НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_004.96_

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

1. Магістерська дисертація

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією – Інженерія програмного забезпечення розподілених систем

на тему_____Розробка архітектури мультиагентної системи_____
___енергетичної інфраструктури_____
_____

Виконав: студент _6_курсу, групи_ТІ-81мп_
_____Міхєєв Олексій Сергійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _доцент, к.т.н._Ковальчук А.М._ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019

Національний технічний університет України
«Київський політехнічний інститут імені
Ігоря Сікорського»

Факультет (інститут) **Теплоенергетичний**
(повна назва)

Кафедра **Автоматизації проектування енергетичних процесів і систем**
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність **121 Інженерія програмного забезпечення**
(код і назва)
спеціалізація **Інженерія програмного забезпечення розподілених систем**

ЗАТВЕРДЖУЮ Завідувач кафедри
_____ _О.В. Коваль_
(підпис) (ініціали, прізвище)

«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Міхєєву Олексій Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації _Розробка архітектури мультиагентної системи енергетичної інфраструктури
науковий керівник дисертації _Ковальчук Артем Михайлович доцент, к.т.н._ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «04» листопада 2019 р. №3813-с

2. Строк подання студентом дисертації _9 грудня 2019р._
3. Об'єкт дослідження стандартизовані архітектурні підходи розробки мультиагентних систем

4. Предмет дослідження програмні засоби розробки архітектури мільтиагентної системи енергетичної інфраструктури

5. Перелік завдань, які потрібно розробити аналіз існуючих стандартів архітектур мультиагентних систем, аналіз сфер застосування мультиагентних систем, визначення можливостей удосконалення існуючих архітектурних рішень,

розробка мови комунікації агентів, розробка архітектури агента

6. Орієнтовний перелік ілюстративного (графічного) матеріалу структури архітектур мультиагентних систем, структури повідомлень мови комунікації агентів, діаграми класів реалізації агентів, діаграми послідовностей комунікації та взаємодії агентів

7. Орієнтовний перелік публікацій Міхєєв О.С. "Децентралізація функцій служби каталогів мультиагентної системи енергетичної інфраструктури з метою підвищення горизонтальної масштабованості та стабільності системи"

8. Дата видачі завдання «19» березня 2019р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	19.03.2019	
2	Збір інформації	20.03.2019–26.03.2019	
3	Аналіз вимог завдання, розробка методів і засобів розв'язання поставленої задачі	27.03.2019–25.06.2019	
4	Розробка та тестування програмного продукту	26.06.2019–21.10.2019	
5	Підготовка матеріалів магістерської роботи	22.10.2019–19.11.2019	
6	Захист програмного продукту	26.10.2019	
7	Передзахист	20.11.2019	
8	Захист	16.12.2019	

Студент _____ О.С. Міхєєв
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ А.М. Ковальчук
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Дисертаційна робота складається зі вступу, 4 розділів, висновків, списку використаних джерел з 30 найменувань. Обсяг дисертації становить 80 сторінок, робота має 20 рисунків, 27 таблиць, 1 додаток.

Актуальність теми. Відкритий характер сучасного інформаційного суспільства та глобальної економіки призводить до загострення конкуренції на ринках і змушує підприємства шукати нові методи і засоби організації і управління, спрямовані на більш якісне і ефективне задоволення індивідуальних запитів споживачів. Один з нових підходів до управління підприємствами пов'язаний з побудовою відкритих мережевих організацій, підрозділи яких, в свою чергу, можуть будуватися і функціонувати як автономні підприємства та безпосередньо взаємодіяти як між собою, так і з зовнішніми організаціями. Така мережева організація дозволяє підприємству більш оперативно отримувати нові знання і гнучко адаптувати свою структуру або принципи функціонування з урахуванням змін в середовищі. Разом з тим, мережева організація призводить до суттєвого ускладнення і інтенсифікації процесів прийняття та узгодження рішень, пов'язаних як з поточним функціонуванням, так і розвитком підприємства. Для того, щоб бути ефективними, такі підприємства повинні постійно здійснювати ідентифікацію потреб і можливостей в середовищі і оперативно змінювати конфігурації своїх виробничих, фінансових та інших ресурси, своєчасно впроваджуючи нові технології, оновлюючи номенклатуру продукції тощо.

Підходом до вирішення даної може бути система, що базується на застосуванні мультиагентих технологій, які отримали інтенсивний розвиток в останніх десятиліттях на стику методів штучного інтелекту, об'єктно-орієнтованого програмування, паралельних обчислень і телекомунікацій. В основі цієї технології лежить поняття «агента» - програмного об'єкта, здатного сприймати ситуацію, приймати рішення і комунікувати з собі подібними, динамічно встановлюючи зв'язки між собою. Ці можливості кардинально відрізняють мультиагентні системи

(МАС) від існуючих систем ієрархічної побудови, забезпечуючи їм таку важливу властивість, як здатність до самоорганізації.

Аналіз існуючих підходів до побудови архітектур мультиагентних систем виявив недоліки у функціональних елементах даних систем. Подібні проблеми призводять до нестабільності системи у критичні моменти та підвищують ризики виходу системи з ладу. Таким чином актуальність роботи полягає у модифікації існуючих стандартів з метою усунення перелічених проблем.

Мета й завдання дослідження. Метою дисертаційної роботи є розробка архітектури мультиагентної системи енергетичної інфраструктури, що може буди модифікована для використання в інших галузях.

Поставлена мета досягається шляхом вирішення наступних завдань:

- Аналіз існуючих підходів до побудови архітектури мультиагентної системи та вияв їх недоліків.
- Аналіз способів використання мультиагентних систем у різних галузях.
- Удосконалення одного з існуючих підходів до побудови архітектури МАС.
- Програмна реалізація основних компонентів удосконаленої архітектури мультиагентної системи.

Об’єкт дослідження. Інформаційні системи, побудовані з використанням мультиагентного підходу, та їх структура.

Предмет дослідження. Програмні засоби реалізації архітектури мультагентної системи енергетичної інфраструктури.

Методи дослідження базуються на використанні методів паралельних обчислень, теорії графів, підходів мультиплатформеної розробки, теорія і методи проектування інформаційних систем, методи об’єктно-орієнтованого програмування для побудови гнучкої структури системи з можливостями розширення.

Інноваційна новизна одержаних результатів полягає в удосконаленні існуючого стандарту архітектури мультиагентних систем FIPA, а саме:

- Запропоновано архітектурний підхід до побудови мультиагентної системи без використання агентів середнього рівня (обслуговуючих агентів).

- Запропоновано архітектуру агента як учасника системи.
- Розроблено інструментальні засоби для побудови мультиагентних систем, що включають засоби комунікації агентів в рамках МАС.

Практичне значення одержаних результатів полягає у розробці програмної інструментальних засобів розробки мультиагентної системи, що можуть бути використані як у рамках енергетичної інфраструктури, так і у інших галузях (логістика, медицина тощо). Розроблені засоби реалізовані мовами програмування C/C++, що дозволяє використовувати їх на більшості сучасних платформ, таких як Raspberry PI, Arduino, STM.

Апробація результатів дисертації. Результати досліджень, включених до дисертації, представлені 43-й міжнародній науковій "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення", м. Тернопіль, 14 листопада 2019р.

Публікації. Основні положення дисертації опубліковані в 1 друкованій роботі.

Ключові слова. МУЛЬТИАГЕНТНІ СИСТЕМИ, АГЕНТ, АГЕНТНИЙ ПІДХІД, FIPA, СЛУЖБА КАТАЛОГІВ, БРОКЕР ПОВІДОМЛЕНЬ, C++.

ABSTRACT

The dissertation consists of an introduction, 4 sections, conclusions, a list of used sources of 30 titles. The dissertation volume is 80 pages, the work has 20 drawings, 27 tables, 1 appendix.

Actuality of theme. The open nature of today's information society and the global economy is exacerbating competition in the markets and forcing businesses to seek new methods and tools of organization and management aimed at better and more efficient customer satisfaction. One of the new approaches to enterprise management is related to the construction of open network organizations, the units of which, in turn, can be built and function as autonomous enterprises and interact directly with each other and with external organizations. Such a network organization allows the enterprise to more quickly acquire new knowledge and adapt flexibly its structure or principles of operation in the

light of changes in the environment. However, the network organization leads to a significant complication and intensification of the processes of making and agreeing decisions related to both the current functioning and development of the enterprise. In order to be effective, such enterprises must constantly identify the needs and opportunities in the environment and promptly change the configurations of their production, financial and other resources, timely introducing new technologies, updating product nomenclature, etc.

An approach to this solution may be a system based on the use of multiagent technologies, which have been intensively developed in recent decades at the intersection of artificial intelligence, object-oriented programming, parallel computing and telecommunications. At the heart of this technology is the concept of "agent" - a software object capable of perceiving the situation, making decisions and communicating with others similar, dynamically establishing relationships with each other. These capabilities fundamentally differentiate multi-agent systems (MACs) from existing hierarchical systems, providing them with such an important property as the ability to self-organize.

The analysis of existing approaches to the construction of architectures of multiagent systems revealed shortcomings in the functional elements of these systems. Such problems lead to system instability at critical times and increase the risk of system failure. Thus, the urgency of the work is to modify existing standards to address these issues.

The purpose and goals. The aim of the dissertation is to develop the architecture of a multiagent system of energy infrastructure, which can be modified for use in other fields.

This goal is achieved by solving the following problems:

• Analysis of existing approaches to the construction of multi-agent system architecture and identification of their disadvantages.

• Analysis of how to use multiagent systems in different industries.

• Improvement of one of the existing approaches to the construction of the MAC architecture.

• Software implementation of the main components of an advanced multi-agent system architecture.

Object. Information systems built using a multiagent approach and their structure.

Subject. Software tools for implementing the architecture of a multi-agent system of energy infrastructure.

Research methods are based on the use of methods of parallel computing, graph theory, approaches of multi-platform development, theory and methods of information systems design, object-oriented programming methods to build a flexible structure of the system with expansion capabilities.

The innovative novelty of the obtained results is the improvement of the existing standard of architecture of multi-agent systems FIPA, namely:

• An architectural approach to the construction of a multi-agent system without the use of mid-level agents (servicing agents) is proposed.

• The agent architecture as a member of the system is proposed.

• Tools have been developed for the construction of multiagent systems incorporating agents' communications tools within the MAC.

The practical significance of the obtained results lies in the development of software tools for the development of a multiagent system that can be used both within the energy infrastructure and in other fields (logistics, medicine, etc.). The developed tools are implemented in C / C ++ programming languages, which allows them to be used on most modern platforms, such as Raspberry PI, Arduino, STM.

Testing the results of the thesis. The results of the research included in the dissertation are presented to the 43rd International Scientific Society "Information Society: Technological, Economic and Technical Aspects of Formation", Ternopil, November 14, 2019.

Publications. The main provisions of the dissertation are published in 1 printed work.

Keywords. MULTI-AGENT SYSTEMS, AGENT, AGENT APPROACH, FIPA, CATALOG OFFICE, MESSAGE BROKER, C ++.

ЗМІСТ

ВСТУП

Вже більше десяти років сучасні міста впроваджують різноманітні системи та програми Internet-of-Things (IoT) у таких сферах, як розумна енергетика, розумний транспорт, побут із підтримкою навколишнього середовища, безпека тощо. Ці програми в більшості випадків розробляються та розгортаються незалежно, що робить їх моніторинг та управління особливо складними. Задля керування різними програмами уніфіковано та з єдиної точки входу, містам доводиться працювати з кількома системами IoT, що включає різні технологічні платформи та пристрої.

Розумні міста являють собою кінцеву візію міського розвитку сучасних міст, яка підтримує міста у протистоянні жорсткому тиску, пов'язаному з урбанізацією, демографічними змінами, зміною клімату та громадською безпекою. У відповідь на цей тиск міста поступово розгортають інфраструктуру та програми IoT. Ці інфраструктури охоплюють різні технології IoT (наприклад, бездротові сенсорні мережі [WSN], радіочастотна ідентифікація [RFID], камери) та націлені на різні сфери застосування, такі як розумна енергетика, розумний транспорт та логістика. Отже, вони схильні утворювати вертикально-фрагментовані силосні програми, якими дуже складно керувати інтегровано, внаслідок необхідності роботи з декількома технологічними платформами та моделями даних для пристроїв, систем і потоків даних IoT.

Завдання моделювання та обчислення стають набагато складнішими, оскільки розмір даних неперервно збільшується. Як результат, централізовані методи обробки даних показують незадовільні за швидкістю результати. Технологія систем на основі агентів протягом останніх років викликала сильний резонанс через її можливості як нової парадигми для концептуалізації, проектування та впровадження програмних систем. Ці можливості можуть бути використані для створення програмного забезпечення, яке працює в умовах реального часу. Наразі, переважна більшість систем на основі агентів складається з одного агента. Однак у міру розвитку технології та вирішенні все більш складних задач стає очевидною потреба у системах, що складаються з декількох агентів, які комунікують між собою.

Система, в якій агенти мають спільний протокол комунікації та використовують його з метою передачі інформації або виконання задач називається мультиагентною (МАС). Принципом МАС є розподіл задач між агентами системи, де кожен агент-учасник системи належить до певної організації (групи). Розподіл задач базується на ролях та можливостях агентів, наявних у системі.

Агенти - це суб'єкти мультиагентної системи, які є автономними, можуть приймати рішення та робити оцінку ситуацій з метою вирішення поставленої задачі. Взаємодія агентів у системі базується на кооперації ресурсів з метою виконання спільної задачі та характеризується розподілом та перевикористанням спільних ресурсів. Розподіл ресурсів є важкою з точки зору машинного часу операцією тому потребує протоколів координації агентів задля економії загальних ресурсів. Координацією агентів є організація послідовності дій між залежними частинами системи. У цей же час в основу координації агентів покладено набір спільних повідомлень та правил їх оптимального та пріоритетного використання. Окрім цього, важливою складовою кооперації агентів є розподіл функцій та виконуваних задач між агентами-учасниками системи.

Мультиагентні системи застосовуються у багатьох областях:

- Управління бізнес-процесами
- Розподілений збір інформації з сенсорів та датчиків
- Пошук інформації та управління нею
- Електронна торгівля
- Людино-машинна взаємодія
- Соціальне моделювання

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Речі в IoT працюють в надзвичайно динамічному середовищі. Вони обмінюються різною кількістю даних з різною швидкістю (наприклад, збільшенням або зменшенням частоти збору даних і передачі сенсорних даних) залежно від їх контексту. Щоб підтримати цей сценарій, вони повинні підтримувати адаптацію апаратних ресурсів, конфігурацію програмного забезпечення та, коли це застосовано, - географічне розташування, щоб впоратись із змінами середовища.

Ці адаптації можуть виконувати оператори людини, але являють собою трудомісткі та схильні до помилок дії. Для того, щоб розкрити потенціал IoT, люди повинні виконати ці адаптації автономно на основі різних критеріїв, включаючи критерії обмінюваних даних.

Загалом, основні компоненти системи IoT можна представити у 3 частинах:

1) Обладнання: апаратна частина означає все обладнання, необхідне для:

• Збору даних та фізичної взаємодії з оточенням, наприклад, датчиками, тегами, камерами тощо.

• Зчитування даних, зібраних з датчиків, та виконання первинної обробки даних.

• Фільтр, синхронізація, агрегація та зберігання даних, таких (термінали та сервери).

2) Програмне забезпечення - відповідає за обробку даних, зібраних з апаратної частини. Це може бути проміжне програмне забезпечення, платформа та операційна система тощо.

3) Протоколи - представляють частину зв'язку між усіма компонентами системи IoT.

Незважаючи на переваги таких систем існує ряд проблем. Серед основних питань, що включають IoT, основними є доступність, сумісність та безпека.

- Доступність. Багато полів, що використовують ІКТ-технології, сильно покладаються на наявність усіх компонентів у системі. Наприклад, перевірка ідентичності за допомогою RFID-зв’язку; якщо зчитувач RFID не працює,

заблокований працівник, ще один серйозний приклад - оплата за допомогою технології NFC, якщо у вашого смартфона не вистачає енергії, ви не зможете пройти транзакцію.

- Взаємодія. Через появу мільйонів пристроїв, підключених у всій мережі, а також проблеми співіснування різних стандартів, багато пристроїв не можуть бути розпізнані з системи в іншу. Тому зв’язок між датчиками і заднім числом буде неможливим.
- Безпека. Безпека викликає серйозне занепокоєння в інфраструктурі IoT. Основна проблема полягає у гарантуванні безпеки компонентів IoT та конфіденційності людей під час усіх процесів обміну даними. Іншими словами, датчики, одиниці обробки даних та сервери повинні зберігатись у конфіденційності, незмінності та доступності.

Для розробки системи IoT ринок програмного забезпечення надає ПЗ декількох типів з різними типами ліцензії. Загалом, ці типи можна класифікувати у 3 категорії:

1. Інструменти прототипування. Коли ви створюєте додаток IoT, інструмент прототипування - це рішення для створення першої невеликої фізичної моделі системи. Насправді цей крок є первинним для перевірки функціональності створеної програми, вимірювання та реалізації тесту на продуктивність, а також аналізу доцільності та зв’язку між компонентами системи. Засоби прототипування, як правило, складаються з апаратної частини, що містить технологічний блок, що взаємодіє з пристроями джерела даних, як датчики, і програмну частину, призначену для розробників для зчитування даних та розробки програми, яка буде розгорнута всередині основного блоку обробки прототипу. У таблиці 1.1 наведено приклад деяких інструментів прототипу IoT.

Таблиця 1.1 – Інструменти прототипування.

Інструмент	Модуль обчислень	Відносна ціна	Характеристики	Середовище розробки	Мови програмування
Arduino [9]	ATmega 328	Низька	Однопоточний, легкий у використанні	Arduino IDE	C/C++

Продовження таблиці 1.1

Raspberry PI [10]	ARM11	Середня	Підтримка Linux, більше різноманіття датчиків, швидка обробка	Python IDE	Python, Java, C/C++
BeagleBone [11]	ARM cortex-A8	Висока	Більша кількість портів, більше обчислювальних можливостей	Cloud9	Java, HTML, CSS

2. Посереднє програмне забезпечення. Посереднє програмне забезпечення розглядається як інтерфейс між бізнес-додатком та апаратним рівнем, якщо в Інтернеті речі, проміжне програмне забезпечення буде виконувати функції оркестру між серверами, які беруть початок, і шаром збору даних. Насправді, середнє програмне забезпечення буде наказувати зчитуючи дані з датчиків, дійте на фізичне середовище, якщо є якісь виконавчі пристрої, фільтруйте та обробляйте зібрані дані, а потім направляйте їх у резервну частину для подальшої обробки. Проблемним тут є поява різних типів проміжного програмного забезпечення щодня з різним рівнем абстрагування програмування та архітектурою доступу до з'єднаних пристроїв IoT [1]. У таблиці 1.2 наведено короткий огляд деяких відомих проміжних програм.

Таблиця 1.2 - Посереднє програмне забезпечення.

ПЗ	Пристрої	Мережеві протоколи	Мови програмування	Віртуалізація	Управління даними
TinyDB [12]	сенсори	HTTP, MQTT	Python	-	збирати/запитувати дані Фільтрування Агрегація Дані маршрутизації Обробка запитів Фільтрація

Продовження таблиці 1.2

GSN [13]		HTTP, UDP, SOAP	Java	+	Обробка запитів Фільтрування Агрегація
Sun Java RFID [14]	RFID	RFID, RMI, XML, JMS, HTTP	Java	-	Центральне адміністрування даних Архівація Фільтрування Агрегація

3. Платформи. Платформа розглядається як глобальний рівень впровадження програми IoT, яка перегрупує всі необхідні інструменти розробника від проектування, розробки, розгортання, зберігання та аналізу даних. Інфраструктура управління даними залишається одним з найбільш критеріїв, які шукають компанії під час вибору відповідної платформи [2]. У таблиці 1.3 показаний приклад повних платформ IOT.

Таблиця 1.3 – перелік платформ IOT

Платформа	Прототипування	Протоколи	Взаємодія	Мови
ThingSpeak[3]	Arduino, Raspberry PI	HTTP, REST	Не визначено	Ruby, Python, Node.js
ThingWonx[4]	Arduino, Raspberry PI	MQQT, DDS, REST	Не визначено	C, Java
KAA [5]	BeagleBone, Raspberry PI	LoraWan, HTTP	Так	C/C++, Java

Для кожної роботи ми враховуємо деякі параметри, які слід враховувати під час процесу складання заявок авторів, враховуючи такі критерії:

• Збір даних: визначає джерело даних, це можуть бути датчики у випадку безпровідної сенсорної мережі, теги у випадку зв'язку із RFID або іншими технологіями.

• Комунікація / протоколи: Як зазначено раніше, багато протоколів підтримуються в процесі обміну даними між шаром збору даних (блок обробки) та резервним рівнем.

• Блок обробки: У разі використання засобу прототипування це поле вказує обладнання, яке використовується для локальної обробки даних.

• Програмне забезпечення: Як стверджується в розділі А), ми визначаємо, чи використовується якась вказана платформа або програмне забезпечення середнього програмного забезпечення для розробки та запуску розробленої програми.

• Відповідність стандарту IoT: Дотримання стандартів є однією з важливих проблем у застосуванні IoT, і саме з цієї причини ми перевіряємо у кожному творі, чи цей пункт був врахований автором.

Перший крок у розумній системі - це отримання даних, що може бути результатом одного компонента зондування, як камера, що використовується у праці [5] для виявлення порожніх місць для паркування або комбінації декількох компонентів збору даних. Використання конкретних протоколів та технологій комунікації виправдано для кожної архітектури системи, контексту використання та компонента, який надсилатиме зібрані дані до блоку обробки. Всі ці протоколи спрямовані на одне первісне посилення, яке полягає у зменшенні енергоспоживання.

Наприклад, автор у праці [6] використовував маяки для виявлення гудів у районі кампусу. Тут близькість обчислюється на основі технології зв'язку BLE, яка забезпечує високу точність оцінки близькості з розумним полем діапазону та низьким споживанням енергії порівняно з NFC та GPS. Це останнє керується та використовується програмним забезпеченням, а точніше середнім програмним забезпеченням, призначеним для об'єднання, обробки, фільтрації та відправлення даних для іншого лікування в локальній чи хмарній мережі. Збір даних з однієї категорії компонентів збору даних може оброблятися спеціальним програмним

забезпеченням процесорного блоку, наприклад, автор [7] використовує ардуїно для взаємодії між датчиками, виконавчими механізмами та зворотним шаром.

Проблема виникає, коли дані надходять із різних джерел компонентів, як це було розроблено у статті [8], в якій автор збирає дані з тегів, камери, датчика відбитків пальців та смарт-картки. Кожна з перелічених у цій роботі технологій використовує власний SDK та інтерфейс для взаємодії між шаром збору даних та резервним шаром, що виправдовує дизайн специфічного середнього програмного забезпечення, яке може синхронізуватися між декількома технологіями.

1.1 Мультиагентний підхід

Комплексні системи складаються з багатьох компонент, що робить оцінку роботи системи досить складної задачею. Основною причиною є мінливість системи – одні компоненти впливають на подальшу роботи інших компонент. Складність подібних систем може зростати експоненційно зі збільшенням кількості компонентів системи. Одним із рішень даної проблеми є декомпозиція цілої системи на підсистеми або використання мультиагентного підходу. Різниця даних підходів наведена на рисунку 1.1.

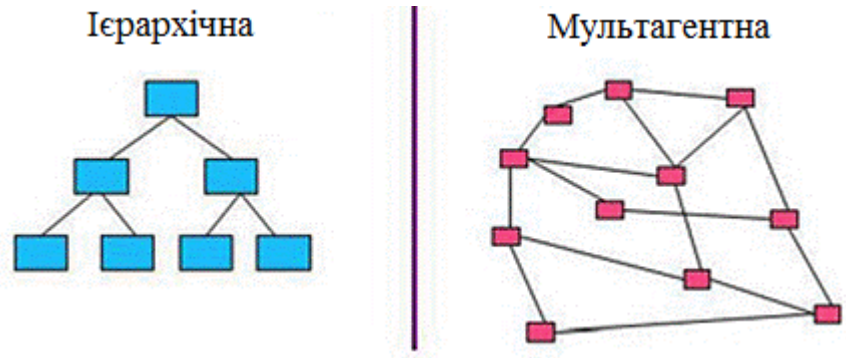

Рис. 1.1. Схеми побудови програмного забезпечення

Керування комплексними системи полегшується за рахунок декомпозиції на підсистеми. Задля досягнення оптимального рівня керування підсистеми оптимізуються індивідуально до оптимального рівня. Розглянемо ієрархічний та децентралізований підходи з метою виявлення переваг.

Основою централізованої системи є агент, який має доступ до усіх знань системи, необхідних для вирішення задач. Ефективність роботи даного агента залежить від кількості доступної йому інформації про систему, що робить обчислення та зберігання такої інформації неефективною задачею. Такий агент є слабким місцем системи і виведення його з ладу робить систему не дієздатною. Окрім цього, головні функції агента такі як обробка та запити інформації можуть сповільнювати швидкість роботи системи. Дана проблема може бути вирішена шляхом розпаралелення задач між учасниками системи шляхом класифікації та поділу інформації між агентами, що є експертами сфер діяльності системи. Експертиза агента – це набір знань та можливих сценаріїв роботи, які агент використовує для вирішення певних задач.

Перевагою використання мультиагентного підходу є можливість проектування системи без чітко визначеної структури з можливістю подальшого вивчення та розширення або адаптування до наступних змін. Однак процес розробки мультиагентних систем не є тривіальним. Окрім цього процес навчання агента не може бути використаний в рамках мультиагентної платформи, так як результати розвитку агента можуть зменшувати ефективність роботи системи або порушувати принципи роботи інших агентів-учасників системи. Дана проблема згадується дослідниками як “Трагедія общин”, де окремі учасники системи неефективно використовують спільні ресурси, розуміючи, що подібні операції знижують ефективність роботи інших учасників у довгостроковій перспективі. Подібні конфлікти потенційно призводять до відмови систем та унеможливлюють використання класичного підходу до навчання Reinforcement Learning (RL). Вирішенням даної проблеми є використання архітектури розподілених систем на основі агентів, кожен з яких є локальним та соціальним, тобто співпрацює з іншими агентами в рамках однієї задачі. Подібні підходи зменшують паралельного обчислення однакової інформації різними частинами системи, що підвищує стабільність та ефективність роботи системи. Використання розподілених та децентралізованих систем на основі мультиагентного підходу вимагає імплементації протоколів та чітких й однозначних правил обміном інформацією.

1.2 Мультагентні системи

Розвиток мультиагентних систем розпочався у 60-х роках 20-го століття. У основу покладено досягнення сфер, нині суміжних до МАС, таких як системи штучного інтелекту, грід-обчислення та обчислення на кластерах (в тому числі й системи паралельних обчислень). Мультиагентні системи поєднують й інтегрують наведені вище сфери з метою автоматизації та оптимізації процесів. У реаліях сучасності, сфера мультиагентних систем є однією з найбільш динамічно розвиваючихся і перспективних напрямків в області штучного інтелекту [15].

Мультиагнетні системи є розширенням технології застосування агентів у середовищі, де група автономних учасників координується з метою досягнення спільної мети. МАС мають ряд переваг через що мають широкий спектр застосування. Розглянемо переваги МАС, наявні при використанні підходу у великих системах:

- Підвищення швидкості та ефективності виконання задач за рахунок розподілених асинхронних обчислень.
- Підвищення надійності системи за рахунок переорганізації агентів. Дана перевага проявляється при виході агентів з ладу.
- Спрощення масштабованості системи, тобто агенти можуть бути додані або виведені з системи за необхідності у будь-який момент часу.
- Зменшення кінцевої вартості системи. Вартість розробки одного агента менша за вартість розробки глобальної архітектури системи.
- Агенти можуть бути перевикористані в рамках однієї системи за рахунок модульності.
- Оновлення окремих агентів легше за оновлення монолітної архітектури.

Таким чином, МАС мають ряд переваг перед одноагентними системами. Однак використання МАС породжує ряд нових проблем. Розглянемо основні критичні проблеми систем.

- Зміни у середовищі. Кожен агент МАС змінює власний стан під час виконання задачі. Окрім цього, можливе змінення стану суміжних агентів. Так як агент повинен передбачати зміни у системі та реагувати відповідно до цих змін, подібна поведінка може призвести до нестабільності системи. Дана проблема наявна у системах, які динамічно змінюється. З метою подолання даної проблеми, кожен агент повинен розрізняти зміни у системі, які викликані його діяльністю, від змін, викликаних роботою інших агентів.
- Робота з сенсорами. Кожен агент має обмежений набір сенсорів. Ліміт задається дальністю діїї цих датчиків та їх площею покриття. У розподілених мультиагентних системах подібні обмеження технічних засобів можуть викликати спотворення інформації, що потенціально призводить до зменшення ефективності роботи системи.
- Абстракція. Використання агентів як частини МАС означає розмежування зон відповідальності між агентами-учасниками системи. Кожен з агентів має уявлення лише про свої функції. Таким чином є можливою конкуренція між агентами з метою досягнення цілі, що зменшує ефективність роботи системи в цілому.
- Конфлікти дій агентів. Дії одного з агентів, можуть призвести до вирішення задачі, але завдадуть негативного впливу на інших агентів-учасників системи. Система намагатиметься зменшити негативний вплив, що призведе до нестабільності внутрішніх станів агентів. Основою проблеми є відсутність інформаціїї про наміри агентів, тому базовий стан агентів повинен передаватися як параметр системи.

1.3 Поняття агента у мультиагентних систем

Дослідники сфери штучного інтелекту не визначили єдиного визначення поняття «Агент». Першою причиною є універсальність даного поняття. По-друге, агенти можуть мати різні фізичні форми від роботів до розподілених комп’ютерних систем та мереж. По-третє, сфери застосування агентів визначаються в залежності

від задачі та не можуть бути узагальнені. Дослідники використовують терміни подібні до softbots (software agents), knowbots (knowledge agents)надали , taskbots (task-based agents) в залежності від області використання агентів [18]. Найбільш поширеним визначенням агента є визначення, які Russell та Norvig. Вони визначили агента як гнучку автономну сутність, що сприймає оточення через власні сенсори. Дане визначення не покриває усіх можливостей агентів.

Агентів можна відрізнити від розподілених або експертних систем за наступними ознаками:

- Поведінка у системі. Алгоритми прийняття рішень агентами не мають залежати від зовнішніх факторів. З іншого боку, агенти можуть взаємодіяти з системою з метою вирішення специфічної задачі, мають змогу надавати інформацію агентам-запитувачам з метою допомоги у вирішенні задачі. Агенти повинні навчатися на основі комунікації з іншими сутностями системи, які можуть бути представлені наступними компонентами: людина, статичні контролери системи, інші агенти системи.
- Ситуативність. Дана властивість описує взаємодію агента з навколишнім середовищем через сенсори агента та через дії інших учасників системи. Агент отримує дані безпосередньо через взаємодію з середовищем чи агентами цього середовища. Агент впливає на середовище через надані середовищем механізми. Подібні властивості агентів відрізняють даний підхід від експертних систем, в яких агент не маю фізичного впливу.
- Автономність. Дана властивість визначається можливістю агента обирати набір дій, необхідних для виконання задачі, без втручань людини або інших агентів-учасників системи. Ця властивість гарантує цілісність та ізольованість внутрішнього стану агента незалежно від зовнішніх факторів.
- Гнучкість. Властивість, що дозволяє агенту швидко реагувати на зміни у середовищі та змінювати внутрішній стан відповідно до цих змін. Дана властивість є головною при побудові додатків, які працюють у реальному часі.

Перелічені властивості не можуть у повній мірі охарактеризувати агента як учасника системи. Додаткові властивості додаються на етапі розробки архітектури системи.

1.4 Сфери застосування мультиагентних систем

У реаліях сучасності використання мультиагнтних систем є можливим як у цілях бізнесу, промисловості, так і на державному рівні. Мультиагентний підхід використовується у невеликій кількості бізнес-сфер [19]:

- Сфера оборони використовує мультиагентні системи у цілях військової підготовки. НАСА використовує подібні системи як основу OCA Mirroring System [20].
- Управління мережами. Використання з метою розподілу та забезпечення безпеки транспортних мереж [21], організація мереж транспортування нафти [22].
- Користувацький інтерфейс, організація взаємодії телекомунікаційних систем. Використання з метою впровадження додаткового рівня між мережевими протоколами та OSS, організація покрокової підтримки [23].
- Логістика, а саме розробка графіків та оптимізація. Системи управління транспортними мережами [24] та оптимізація подачі тепла під час виробництва сталевих продуктів [25].
- Керування системами управління у промисловості. МАС використовується під час планування серійного виробництва двигунів Skoda Auto [26].
- Імітаційне моделювання з метою навчання керівників. Наприклад, компанія EuroBios використовує МАС з метою уточнення графіків виробництва картонних коробок [27].

Мультиагентний підхід може бути використаний у широкому спектрі галузей та може вирішувати набір проблем кожної з них. Окрім цього, використання систем на основі МАС робить систему більш простою для підтримки та розвитку. Проектування систем на основі агентів є більш простим з точки зору бізнес-

аналітиків, аніж використання інших архітектурних підходів. Окрім цього доцільно здійснити модифікацію існуючих систем від орієнтації на об’єкт на орієнтацію на агент [28].

1.5 Висновки до розділу 1

Недоліком централізованого підходу є неможливість використання подібних систем у реаліях сучасних динамічних середовищ, так як обчислення нової інформації може бути запитане під час некоректного стану системи, що зменшує швидкість обробки інформації. З іншого боку, декомпозиція подібних систем може підвищити ефективність роботи системи за рахунок взаємодії окремих частин-підсистем. Автономні агенти можуть виконувати обчислення незалежно від стану інших агентів-учасників системи та надавати результати паралельно до оброки. Набір таких агентів підвищує гнучкість системи та можливості її адаптації до запитів користувачів. Кооперація агентів відіграє важливу роль у рамках мультиагентних систем, так як підвищує швидкість та якість обробки інформації. Перевагою мультиагентного підходу є можливість кооперації агентів з метою виконання спільної задачі та можливість колективного навчання агентів з метою виявлення проблем задачі.

2. ІСНУЮЧИ ТА ВЛАСНІ МЕТОДИ РЕАЛІЗАЦІЇ

Мультиагентні системи (МАС) розробляються з використанням різних архітектур. Вони варіюються від “peer-to-peer” до “blackboard-like”, від небезпечних розподілених систем до складних схем безпеки, від тих МАС, що використовують бази даних, до тих, що використовують інформаційні системи на основі знань.

Компоненти відкритої архітектури повинні забезпечувати більш високий рівень пристосованості, мобільності та інтелекту, щоб підтримувати додатки, які потребують функціонування у змінних умовах. Наприклад, компонентам, можливо, знадобиться взаємодіяти з іншими компонентами, невідомими під час реалізації компонента. Компонент може мати цілі відкритого типу, такі як дослідницькі компанії, які продають мікросхеми пам'яті. Для цього компоненту може знадобитися виявлення та спілкування з іншими компонентами, які мають ці знання.

Розглянемо основні стандарти, які використовуються під час розробки архітектури мультиагентних систем.

2.1 Semantic Web Services Architecture

Веб-сервіси надають стандартні засоби взаємодії між різними програмними програмами, що працюють на різних платформах та / або структурах. WSA пропонує концептуальну модель та контекст для розуміння веб-служб та взаємозв'язку між компонентами цієї моделі.

Архітектура не намагається вказати, як реалізуються веб-сервіси, і не накладає обмежень на поєднання веб-служб. WSA описує як мінімальні характеристики, спільні для всіх веб-служб, так і ряд характеристик, які потрібні багатьом, але не всім, веб-сервісам.

Архітектура веб-служб - це архітектура інтероперабельності: вона ідентифікує ті глобальні елементи глобальної мережі веб-служб, які необхідні для забезпечення сумісності між веб-службами.

Веб-сервіс - це абстрактне поняття, яке має реалізовувати конкретний агент. Агент - це конкретна частина програмного чи апаратного забезпечення, яка надсилає та приймає повідомлення, в той час як послуга - це ресурс, який характеризується абстрактним набором функціональних можливостей, який надається. Щоб проілюструвати цю відмінність, ви можете реалізувати певний веб-сервіс, використовуючи один день одного агента (можливо, написаного однією мовою програмування), а іншого агента наступного дня (можливо, написаного на іншій мові програмування) з однаковою функціональністю. Хоча агент, можливо, змінився, веб-служба залишається такою ж.

Метою веб-сервісу є надання певної функціональності від імені її власника - людини чи організації, наприклад, бізнесу чи фізичної особи. Організація-постачальник - це особа чи організація, яка надає відповідного агента для впровадження певної послуги. Суб'єкт, що вимагає запиту - це особа чи організація, яка бажає скористатися веб-послугою суб’єкта господарювання. Він використовуватиме агент-запитувач для обміну повідомленнями з агентом постачальника суб'єкта господарювання.

Механіка обміну повідомленнями задокументована в описі веб-сервісу (WSD). WSD - це специфікація інтерфейсу веб-служби, записана WSDL. Він визначає формати повідомлень, типи даних, транспортні протоколи та формати серіалізації транспорту, які слід використовувати між агентом-запитувачем та агентом постачальника. Він також вказує одне або кілька мережевих місць, в яких може бути викликаний агент провайдера, і може надати деяку інформацію про шаблон обміну повідомленнями, який очікується. По суті, опис послуги являє собою угоду, що регулює механіку взаємодії з цією службою.

2.1.1 Основні компоненти

Архітектура оперативної сумісності SWSA охоплює такі класи функцій підтримки, які виконуються агентами Semantic Web (постачальниками послуг, запитувачами та середніми агентами). Хоча не у всіх операційних середовищах буде потрібно підтримувати всі функції в однаковій мірі, передбачено, що розподілені функції, на які повинна відповідати ця архітектура, включатимуть:

- Динамічне виявлення сервісу: здатність програмного агента шляхом взаємодії з іншими агентами виявляти послуги-кандидати для конкретних цілей.
- Залучення послуг - переговори та укладання договорів: здатність двох агентів взаємно формувати взаємодію в режимі офлайн або онлайн, спільну угоду про умови надання послуги, яку повинен надавати один агент для іншого. Сюди входить публікація моделей послуг, які послуги зобов’язуються дотримуватися, та більш динамічні взаємодії, що призводять до контрактів між клієнтами та службами.
- Увімкнення та управління процесом обслуговування: можливість динамічного вибору та виклику або взаємодії з послугами для досягнення певної визначеної мети. Сюди входить формулювання запитів на обслуговування, що відповідають усім семантично описаним критеріям прийняття, інтерпретація всіх повідомлень від постачальників послуг, а також моніторинг стану виконання критеріїв та критеріїв завершення послуги, погоджених на контракті. Якщо визначено, він також включає можливість інтерпретувати та вводити пов'язані механізми скасування, відновлення відмов та компенсації. Введення в дію процесу обслуговування може вимагати дотримання розширених протоколів та планів взаємодії, які передбачають безліч служб. Плани можуть бути самостійно створені або опубліковані семантичною мовою процесу та інтерпретовані.
- Послуги підтримки семантичної веб-спільноти: Можливості, пов’язані з обміном семантичними описами, онтологіями та онтологічними відображеннями та каталогами сервісів у межах та поміж спільнотами. Підтримка управління членством у громаді, конфіденційності та відносин влади та спільних інформаційних ресурсів.
- Послуги життєвого циклу та управління ресурсами Semantic Web: Можливість нерестувати, розподіляти, керувати та звітувати про життєвий стан служб, обчислювальних хост-платформ та сховищ для веб-сервісів Semantic.
- Наскрізні проблеми: Семантичні уявлення про два види мета-властивостей якості послуг можуть бути важливими під час виявлення відповідних послуг, можуть з’являтися в угодах про обслуговування та можуть контролюватися під час виконання. Перший набір властивостей стосується якості виконання послуг,

швидкості, своєчасності, повноти результату та ефективності, які можуть зробити певні послуги більш-менш придатними для використання певними клієнтами або в певних контекстах. Другий клас властивостей представляє політику послуг щодо безпеки, конфіденційності та довіри.

Архітектура описується у вигляді декількох простих елементів: концепцій, відносин та моделей. Поняття часто є іменниковими, оскільки вони ідентифікують речі чи властивості, які, як ми очікуємо, побачимо в реалізації архітектури, аналогічно, це стосунки, як правило, мовні дієслова. Як і у випадку будь-яких масштабних зусиль, часто потрібно структурувати саму архітектуру. Модель - це цілісна частина архітектури, яка фокусується на певній темі чи аспекті архітектури.

Очікується, що концепція матиме певну відповідність будь-яким реалізаціям архітектури. Наприклад, концепція повідомлення ідентифікує клас об'єктів (не плутати їх з Об'єктами та класами, як це можна знайти в об'єктно-орієнтованих мовах програмування), який, як ми очікуємо, зможе ідентифікувати в будь-якому контексті веб-служб. Точна форма повідомлення може бути різною в різних реалізаціях, але концепція повідомлення підказує нам, на що звернути увагу в конкретній системі, а не наказувати її точну форму.

Не всі концепції матимуть реалізацію з точки зору об'єктів даних або структур, що виникають в комп'ютерах або комунікаційних пристроях; наприклад, людина чи організація стосується людей та людських організацій. Інші поняття все ж більш абстрактні; наприклад, надійність повідомлення позначає властивість послуги транспортування повідомлень - властивість, яку не можна торкатися, але тим не менш важливо для веб-служб.

Кожна концепція представлена в регулярному стилізованому вигляді, що складається з короткого визначення, перерахунку взаємозв'язків з іншими поняттями та трохи більшого пояснювального опису. Наприклад, поняття агента включає як пов'язані поняття той факт, що агент є обчислювальним ресурсом, має ідентифікатор і власника.

Модель - це цілісна підмножина архітектури, яка, як правило, обертається навколо певного аспекту загальної архітектури. Хоча різні моделі поділяють поняття

з різних точок зору; головна роль моделі полягає в поясненні та інкапсуляції значної теми в загальній архітектурі веб-служб.

Наприклад, модель, орієнтована на повідомлення, фокусується та пояснює веб-сервіси строго з точки зору проходження повідомлення. Зокрема, вона не намагається пов’язати повідомлення з наданими послугами. Однак модель, орієнтована на сервіс, лежить на вершині і розширює модель, орієнтовану на повідомлення, щоб пояснити основні поняття, що беруть участь у сервісі - фактично для пояснення мети повідомлень у моделі, орієнтованої на повідомлення.

2.2 The Foundation for Intelligent Physical Agents

The Foundation for Intelligent Physical Agents (FIPA) - це спільнота розробників, метою яких є стандартизація агентних технологій. Результатами роботи даної спільноти є набір специфікацій та стандартів розробки та використання МАС. Основними специфікаціями є стандарт управління агентами та стандартизована мова комунікації агентів (Agent Management - AM та Agent Communication Language – ACL).

Управління агентам забезпечується використанням системи, яка інкапсулює взаємодію агентів-учасників. Система визначає правила створення, реєстрація, взаємодії (комунікації та кооперації), локації, міграції та виведення агентів з системи.

Модель управління агентами складається з наступних компонентів, кожен з яких представляє набір можливостей. Розглянемо основні компоненти архітектури управління FIPA

Агент - це автономний обчислювальний процес, який наділений комунікативної функціональністю в рамках системи. Комунікативність досягається за рахунок обміну інформацією та основана на ACL. Агент є основною сутністю МАС та описується набором сервісів та функцій, які описуються агентною моделлю.

Сервіс Каталогу (Directory Facilitator (DF)) - це опціональний компонент Агентної Платформи, але якщо він присутній, то він реалізується як спеціальна

служба-куратор. Служба Каталогу містить інформацію про можливості інших агентів (так звані «жовті сторінки»). Служба Каталогу повинна надавати найактуальнішу інформацію про стан системи усім авторизованим агентам-учасникам. Кожен агент, що публікує свої послуги через Службу Каталогів, повинен зареєструвати опис своїх можливостей у відповідній службі. При цьому Служба Каталогів не визначає валідність інформації, наданої агентом, та не контролює життєвий цикл агентів.

Система Управління Агентами є головним та обов'язковим компонентом мультиагентної системи. Лише одна сутність Системи Управління може бути присутня у МАС. Головними функціями системи управління є контроль операцій створення та видалення агентів, а також їх міграції з МАС, за умови, що дана МАС підтримує міграцію агентів. На відміну від Служби Каталогів, Система Управління Агентами має актуальну інформацію про стан агентів даної системи. Система Управління Агентами відповідає за підтримку списку усіх агентів, що є учасниками даної системи. Кожен агент реєструється в МАС, надсилаючи власну унікальну інформацію до Системи Управління, та отримує унікальний токен-ідентифікатор у результаті даної процедури.

Транспортна Служба – складова МАС, що відповідє за передачу повідомлень агентів в рамках однієї МАС. Усі агенти FIPA повинні мати доступ хоча б до однієї транспортної служби МАС. Обов'язковою умовою користування Транспортною Службою є наявність кінцевого адресата-агента у повідомленні.

Агентна Платформа (Agent Platform (AP)) представляє фізичну інфраструктуру, в якій можуть бути розгорнуті агенти. Окрім FIPA-агентів та служб, перелічених вище, Агентна Платформа має наступні компоненти: програмне забезпечення, необхідне для агентів, операційна система та обладнання, що використовується для роботи агентів та системи в цілому. Структура Агентної Платформи не входить до стандартів FIPA та може базуватися на інших архітектурних підходах. Концепція Агентної Платформи дозволяє використовувати різні підходи до її організації, такі як агентні потоки, розподілені мережі тощо.

Стандарти FIPA надають базові визначення понять комунікації агентів в рамках однієї Агентної Платформи або між декількома платформами або МАС за умови, що платформи підтримують однаковий набір протоколів комунікації.

В архітектурі мультиагентних систем можна виділити наступні компоненти та поняття:

- Архітектура агента – описує структуру агента, його комунікаційні можливості, соціальну активність та цілі агента в рамках Агентної Платформи.
- Архітектура агентної системи – описує можливих варіантів взаємодії агентів, як сутностей МАС, визначає основні правила щодо оптимізації ефективності роботи агентів.
- Фреймворк агентів - набір програмних засобів, які використовуються під час створення агентів. Прикладами таких засобів є Jade, Voyager;
- Інфраструктура – набір правил, які описують принцип взаємодії агентів між собою, з метою кооперації в рамках поставленої системою.

Агентна інфраструктура описує наступні поняття:

- Онтологія - опис основних понять та концепцій системи;
- Протоколи комунікації – набір повідомлень комунікації агентів;
- Інфраструктура комунікацій – канали та способи комунікації агентів;
- Протоколи взаємодії – набір угод та поведінок взаємодії агентів.

Архітектура визначає ряд механізмів інформування та пошуку агентів; об'єднання, використання, управління та оновлення інформації про агентів і їх активності. Виконання перерахованих дій є обслуговуванням агентів системи, що виконується агентами середнього рівня. Агенти середнього рівня надають інформацію, необхідну для виконання задач інших агентів. Перевагою даного підходу є підвищення стійкості системи до змін щодо вводу. переміщення та виведення агентів-учасників системи.

FIPA передбачає наступні агенти, що можуть виступати як агенти середнього рівня:

- Агент-куратор - агент, який координує активність інших агентів і виконує запити своїх підлеглих агентів.

- Агент-посередник - агенти, які використовують знання для створення служб додатки більш високого рівня.
- Агент-брокер - агент, який виконує задачу за рахунок ресурсів агентів-учасників системи.
- Пошуковий агент - агент, який використовують транспортні служби з метою пошуку інших агентів-служб.
- Дошка оголошень – набір агентів, що зберігають запити, отримані в ході роботи системи.

Агенти FIPA характеризуються можливістю взаємодії в рамках Агентної Платформи. Метою взаємодії агентів може бути поширення інформації про зміни станів системи, запити на виконання задач тощо. Основою взаємодії агентів є наступні поняття:

- Мова комунікації агентів (ACL), у тому числі набір повідомлень системи;
- Протоколи комунікації.
- Онтології.

2.2.1 Характеристики МАС

Мультиагентним системам характерно наступне:

- кожен агент має незавершену інформацію, або можливості для вирішення проблеми, таким чином, кожен агент має обмежену точку зору;
- система не має єдиної точки контролю;
- дані, які використовують агенти, є децентралізованими;
- виконання задач відбувається паралельно та асинхронно.

Спостерігається зростання інтересу, що відноситься до дослідження МАС. Причини, за якими це відбувається, такі: МАС здатні забезпечити стійкість проти помилок і ефективність; МАС дозволяють встановлювати зв'язок із існуючим спадщиною систем; МАС здатні вирішувати завдання, в яких управління, дані, експертні знання є розподіленими.

2.2.2 Координація в мультиагентих системах

Питання координації агентів у системі є головною проблемою розробки архітектури МАС. Агенти не виконують лише власну задачу в рамках МАС, а кооперуються з іншими учасниками системи з метою виконання задачі. Кооперація агентів вимагає синхронізації дій, що забезпечує стабільність роботи системи. Координація агентів оптимізує роботу системи що зменшує час виконання задачі. Основними вимогами до координації агентів є:

- Виключення можливості хаотичних дій агентів у системі;
- Збір а поширення інформації та інших спільних ресурсів між агентами;
- Виключення конфліктів дій агентів під час виконання задачі;
- Підвищення ефективності системи;

Координація досягається обмеженням спільних дій агентів або обміном інформацією між пов’язаними агентами.

Дослідники розглядають три основні складові комунікації агентів:

- Протокол комунікації - набір правил, які регулюють взаємодією агентів. Вони визначають, допустимі типи агентів-учасників, можливі стани та події, що викликають переходи в інші стани;
- Предмет комунікації - набір задач, які мають бути виконані, та допустимі операції під час виконання задач. Складність предмету як сутності може варіюватися в залежності в умов системи. Допускається використання як простих дій типу підтвердження, так і багатоступеневих дій, що характеризуються гнучкістю системи;
- Агентна модель – набір дій та опис зв’язків між ними, що утворює логіку вирішення задачі, над якою може працювати агент. Модель залежить від складності протоколів та предметів, що використовує агент.

Недоліком системи, що базується на протоколах комунікації є необхідність реалізації та використання агента, який відповідє за коопреацію. Подібна стратегія комунікації є пасивною, тому що кожен агент має власну реакцію на зміни системи і не може обрати нав’язану стратегію поведінки. Дана стратегія реалізується за допомогою ітеративних послідовних комунікації мж агентами-учасниками зі змінними предметами комунікацій.

Модель FIPA є прикладом агентної платформи, що уніфікує стандарти протоколів комунікації агентів з метою їх ефективної координації. Стандарт FIPA має власну мову комунікації агентів, що описує низькорівневу взаємодію агентів.

2.2.3 Повідомлення мови комунікації FIPA

Взаємодія і комунікація агентів базується на поширенні інформації в МАС за допомогою повідомлень. Існують різні стандарти побудови мов комунікації агентів. Найпоширенішими є KQML та FIPA. Загалом ці мови схожі за концепціями та принципами їх застосування. Відмінністю їх є семантичні структури повідомлень.

Мова комунікації агентів (Agent Communications Language) – це формат повідомлень та протоколів, що використовуються для обробки цих повідомлень з метою оновлення знань агентів системи до актуального стану. Мова комунікації FIPA має 3 рівні:

1. Рівень комунікації. Даний рівень є найнижчим та описує основні параметри, необхідні для доставки повідомлень (адреси отримувача та відправника, унікальні ідентифікатори).

2. Повідомлення. Даний рівень описує можливості мови та використовується як набір правил взаємодії агентів.

3. Вміст. Даний рівень містить онтології, необхідні для виконання задач агентами.

Повідомлення мови FIPA має наступний вигляд (рисунок 2.1):

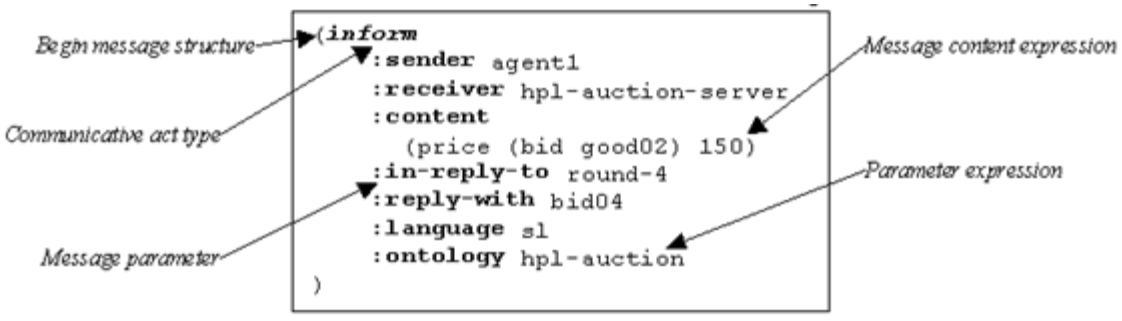

Рисунок 2.1 – Структура повідомлення стандарту FIPA

Повідомлення містить набір ключових слів, що визначають основні параметри, такі як задача або дія, адреси та інформація про отримувача тощо.

Розглянемо основні ключові слова, що присутні в базових повідомлення FIPA:

- “inform”. Дане ключове слово визначає дію. Будь-яке повідомлення FIPA починається зі слова, що хаарактеризує задачу.
- “sender” – блок даних, що містить інформацію про відправника (адреса та ідентифікатори).
- “receiver” – блок даних, що містить інформацію про отримувача (адреса та ідентифікатори).
- “content” – інформація, що передається повідомленням.
- “language” – мова, яку слід використати для розкодування повідомлення.
- “ontology” – онтологія для інтерпретації. Функції онтологій розглянемо у наступному розділі.
- “reply-to” – блок даних, що містить інформацію агента, який повинен отримати відповідь (адреса та ідентифікатори).
- “in-reply-to” – унікальний ідентифікатор, що описує приналежність повідомлення до конкретного діалогу агентів.

2.2.4 Онтології FIPA

Онтологія - це формальне подання понять і аксіом, що регулюють певну область знань. Визначення понять та аксіоми, що обмежують використання цих визначень, становлять онтологію. Онтологія формує теоретичну основу для зазначеної доменної моделі в рамках обмежень даним доменом. Наприклад, Мова специфікації процесів [29] (PSL), розроблена в Національному інституті стандартів і технологій, є онтологією виробничих процесів. Онтології були розроблені у великих масштабах з метою представлення загальних понять, наприклад, Cyc KB [30] представляє факти та евристику для міркувань про предмети та події повсякденного життя. Онтології також розроблені для конкретних областей та дисциплін. Обсяг онтології визначає зручність її використання. Програмні засоби, такі як браузери, редактори та валідатори, часто доступні з певними онтологіями домену, щоб зробити онтології придатними для цього домену.

Атрибут онтології необхідний і вказує посилання на вміст для ACL на платформі агента. Усі агенти платформи, які обмінюються повідомленнями, повинні посилатися на загальну модель даних. Сукупність понять, що містяться в моделі

даних, становить онтологію для перформативного ACL. Онтологія може бути більш-менш формально виражена, з формальним виразом, який вказує на те, що даються формальні визначення понять, а також аксіоми, що обмежують їх використання. Якщо онтологію просто оголосити без формальних визначень, що даються поняттям, агенти все одно можуть обмінюватися повідомленнями, якщо вони посилаються на ту саму онтологію. Однак сумісність з іншими агентами або з іншими платформами вимагає перекладу між різними онтологіями. Це буде ґрунтуватися на спеціальних рішеннях розробника, якщо онтології формально не будуть виражені, оскільки в цьому випадку немає офіційних визначень для обміну.

2.3 Власна реалізація

Дана робота є удосконаленням стандарту FIPA. Як зазначено у попередньому підрозділі, FIPA архітектура має певний набір основних компонент, які забезпечують роботу системи. Проаналізувавши структуру архітектури FIPA та існуючі реалізації, наступні компоненти є найбільш вразливими: служба каталогів, брокер повідомлень.

2.3.1 Модифікація служби каталогів

Служба каталогів - це компонент мультиагентної платформи, який надає агентам послуги з жовтими сторінками. Основною метою служби каталогів є надійне збереження інформації щодо стану агентів активної системи, а також підтримання точного, повного та актуального переліку агентів. Тобто, служба каталогів надає найактуальнішу інформацію про будь-якого агента свого каталогу на недискримінаційній основі всім уповноваженим агентам.

Кожен агент, який хоче оприлюднити свої послуги іншим агентам, повинен знайти відповідну службу каталогів та запросити реєстрацію його опису. Для реєстрації необхідно надати опис агента, що містить обов'язкові параметри опису. Також можуть бути надані необов'язкові та приватні параметри, що містять стандартизовану інформацію, яку розробник агента вважає необхідною до збереження у каталозі. Не передбачено майбутніх зобов'язань чи зобов'язань з боку

зареєстрованого агента. Наприклад, агент може відмовити в запиті на виконання функції, яка зареєстрована через «Службу каталогів». Крім того, служба каталогів не може контролювати життєвий цикл будь-якого з зареєстрованих агентів.

Наявність каталогу агентів і єдиного головного контейнера платформи дозволяє агентам спілкуватися (обмінюватися інформацією) між собою. При цьому така взаємодія можливо всередині контейнера, між контейнерами однієї платформи (між агентами різних контейнерів на одній платформі) і між платформами (між агентами в контейнерах різних платформ). Обмін повідомленнями здійснюється асинхронно, з використанням черг.

Служба каталогів є найбільш навантаженим агентом мультиагентної системи через постійні звернення за інформацією з боку інших учасників системи. Стандарт FIPA передбачає наявність декількох служб каталогів з метою зменшення навантаження на кожен з каталогів. Такий підхід збільшує час обробки інформації з метою прийняття рішень з боку функціональних агентів, що може призвести до непередбачуваних наслідків у рамках енергетичної інфраструктури.

Одним із варіантів розподілу навантаження на службу каталогів є децентралізація функцій реєстрації та дереєстрації між агентами-учасниками системи. Кожен агент містить інформації про свої функціональні можливості та локально зберігає інформацію лише про необхідних йому агентів або служб. Використання протоколу UDP дозволяє зменшити залежність між агентами та обмінюватись інформацією про актуальний стан системи без встановлення з’єднання за допомогою широкомовних повідомлень.

Даний підхід породжує проблему комунікації агентів у розподілених мережах. Так як UDP повідомлення можуть бути відправлені в рамках однієї підмережі, необхідно взяти до уваги можливість використання мультиагентної системи на підприємствах з декількома мережами.

Для передачі подібних повідомлень між локальними підмережами або глобальними мережами є можливість організувати агент типу проксі, метою та єдиною функцією якого буде передача будь-яких повідомлень між агентами

подібного типу. Для цього необхідно модифікувати існуючу мову обміну повідомленнями додавши нові типи задач.

Іншим можливим вирішенням даної проблеми може бути використання технології IP multicast. Багатоадресна передача - це техніка для спілкування в режимі реального часу через IP-інфраструктуру в мережі. Він масштабує більшу кількість одержувачів, не вимагаючи ані попереднього знання про особу одержувача, ані попереднього знання кількості приймачів. Multicast ефективно використовує мережеву інфраструктуру, вимагаючи від джерела надсилати пакет лише один раз, навіть якщо його потрібно доставити великій кількості приймачів. Вузли в мережі (зазвичай мережеві комутатори та маршрутизатори) виконують реплікацію пакета, щоб дістатись до декількох приймачів, таким чином, що повідомлення надсилаються по кожній ланці мережі лише один раз.

Найпоширеніший протокол транспортного рівня для використання багатоадресної адресації - це User Datagram Protocol (UDP). За своєю природою UDP не є надійним - повідомлення можуть бути втрачені або доставлені поза порядком. Надійні протоколи багатоадресної передачі, такі як Pragmatic General Multicast (PGM), були розроблені для додавання виявлення втрат та їх повторної передачі поверх IP багатоадресної передачі.

2.3.2 Модифікація брокеру повідомлень

Реалізація архітектури FIPA передбачає можливість введення додаткового агента-брокера, задачами якого є передача повідомлень до одержувача або збереження повідомлень доки одержувач не зареєструється в мережі. Використання агента такого типу вимагає додаткових заходів щодо забезпечення безпеки системи та підвищення стійкості системи.

2.4 Висновок до розділу 2

Даний розділ висвітлює основні стандарти розробки архітектури MAC.

Запропоноване покращення вирішує наступні проблеми існуючих архітектур мультиагентних систем:

- зменшення часу на прийняття рішення за рахунок збільшення швидкості отримання інформації про агента-виконавця
- зменшення залежності агент-сервіс за рахунок використання UDP повідомлень, що передбачає також зменшення витрат на організацію та підтримку системи
- зменшує необхідність у вертикальному масштабуванні системи адже кожен агент одночасно є сервісом каталогів.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Як зазначено у розділі 2, дана архітектура є удосконалення стандарту FIPA. У рамках дослідження було виявлено, що служба каталогів та брокер повідомлень є слабкими місцями. Розглянемо яким чином з програмної точки зору інтегровано службу каталогів до структури агента.

3.1 Загальна архітектура агента

Кожен агент є автономним та не потребує додаткових агентів або сторонніх служб для виконання базових функцій. Під базовими функціями будемо розуміти наступні: ідентифікація в мережі, обмін повідомленнями та виконання задач. Базові функції агента описані на діаграмах 3.1, 3.2 та 3.3.

Рисунок 3.1 – Базові функції, ідентифікація в мережі.

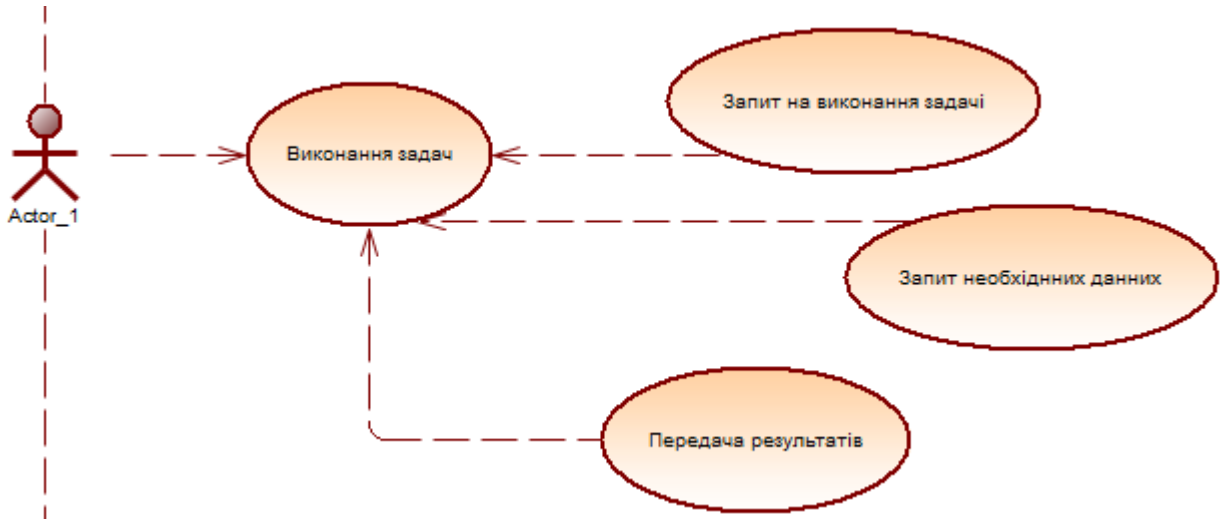

Рисунок 3.2 – Базові функції, виконання задач.

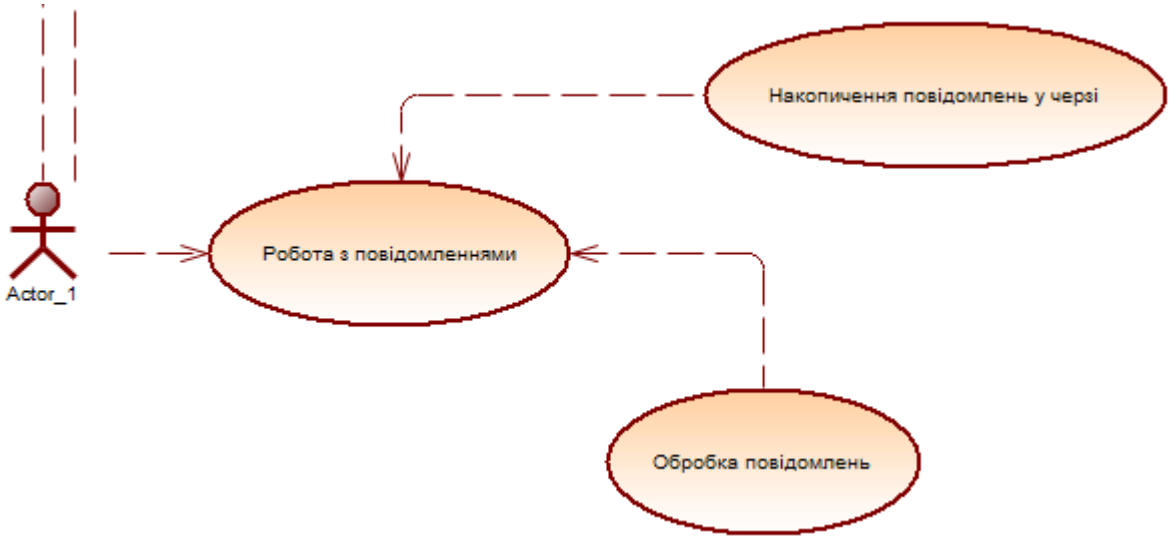

Рисунок 3.3 – Базові функції, робота з повідомленнями.

Кожен агент складається з модулів, незалежних один від одного. Основними модулями є служба каталогів, транспортна служба та набір інтерфейсів для логіки агента. Спрощена архітектура наведена на рисунку 3.4.

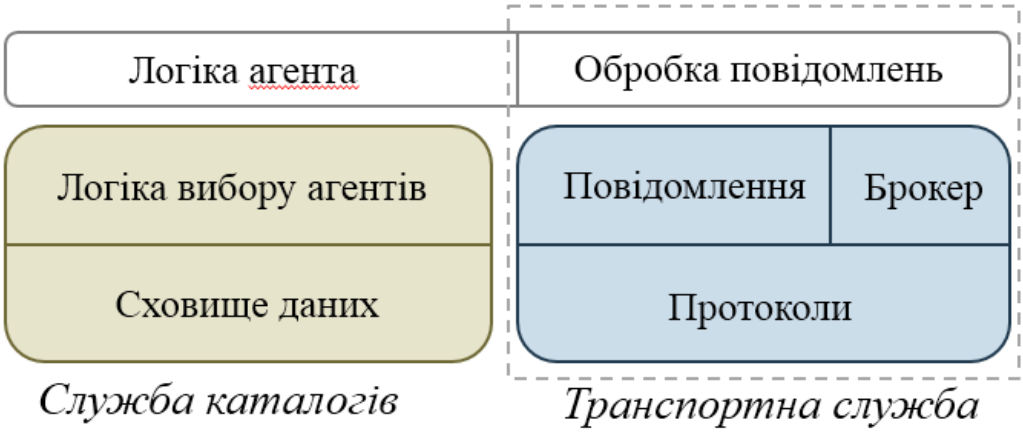

Рисунок 3.4 – Структура агента

Для реалізації архітектури використанці принципи інверсії управління (Inversion of Control) та впровадження залежностей (Dependency Injection) з метою зменшення зв’язності коду та модулів. Інверсія управління - це абстрактний принцип, набір рекомендацій для написання слабо пов'язаного коду. Суть принципу в тому, що кожен компонент системи повинен бути якомога більш ізольованим від інших, не покладаючись в своїй роботі на деталі конкретної реалізації інших компонентів. Впровадження залежностей - це стиль налаштування об'єкта, при якому поля об'єкта задаються зовнішньої сутністю. Іншими словами, об'єкти налаштовуються зовнішніми об'єктами.

Даний підхід дозволяє модифікувати конкретні модулі виключаючи потребу модифікації системи в цілому. Таким чином, розробник агента може замінити способи збереження даних на власну реалізацію тощо.

3.2 Служба каталогів

За стандартом FIPA служба каталогів виконує роль «жовтих сторінок», що містять інформацію про агентів та послуги, які вони надають. В рамках даної модифікації, служба каталогів є однією зі складових частин базової структури агента. Такий підхід дозволяє виключити службу каталогів як окремого агента з архітектури, оскільки він є слабкою ланкою та потребує значних ресурсів задля функціонування системи.

Імплементація служби каталогів складається з двох основних частин: логіка вибору агентів, необхідних для виконання задачі, та сховища даних.

3.2.1 Логіка вибору агентів

Наразі, кожен агент має максимально просту логіку вибору. Останній зареєстрований і доступний у мережі агент може і буде використаний з метою виконання задачі. Якщо агент не є доступним у мережі, він видаляється зі списку. Таким чином, кожен агент зберігає актуальний стан системи та надсилає запити лише до активних частин системи.

Загальний вигляд інтерфейсів наведений на рисунку 3.5.

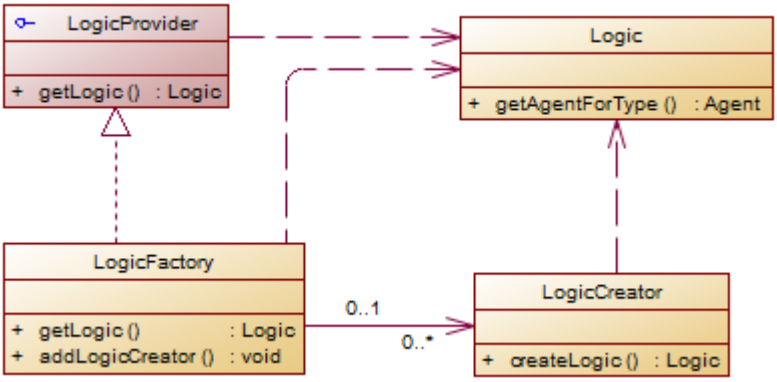

Рисунок 3.5 – створення логіки вибору

Використання інтерфейсів дозволяє замінити фабрику на статичну логіку з метою тестування модуля. Розробник агента може додати власну логіку вибору з мінімальними змінами у системі.

3.2.2 Сховище даних

Кожен агент зберігає в оперативній пам’яті актуальні дані про стан мережи. Доступ до такого типу пам’яті є найбільш швидким тому дозволяє зменшити час обробки задач та прийняття рішень. Однак такий підхід ускладнює відладку програми. З метою вирішення даної проблеми може бути додана можливість збереження стану у базу даних sqlite. Одним з найбільш часто використовуваних патернів при роботі з даними є патерн 'Репозиторий'. Репозиторій дозволяє абстрагуватися від конкретних підключень до джерел даних, з якими працює агент, і є проміжною ланкою між класами, які безпосередньо взаємодіють з даними (рисунок 3.6).

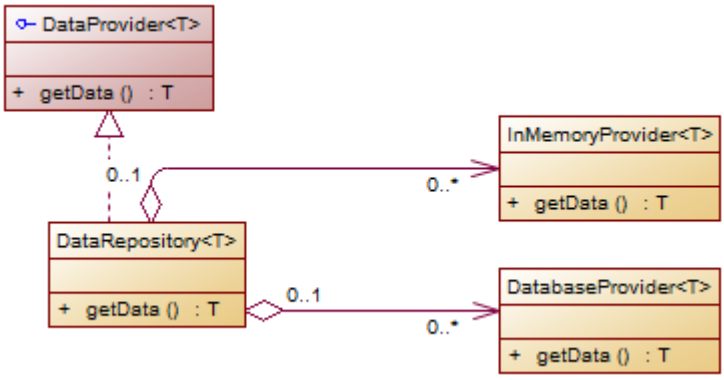

Рисунок 3.6 – Використання репозиторію.

3.3 Транспортна служба

За стандартом FIPA, референтна модель транспортування повідомлень агента включає три рівні:

1. Протокол транспортування повідомлень (MTP) використовується для фізичної передачі повідомлень між двома агентами.

2. Служба транспортування повідомлень (MTS) - це послуга, що надається агентною платформою (АП), до якої приєднаний агент. MTS підтримує транспортування повідомлень між агентами на будь-якій заданій АП та між агентами на різних АП.

3. Мова комунікації представляє корисне навантаження повідомлень, що передаються як MTS, так і MTP.

Дана реалізація розширює рівень №2 з метою відмови від агента брокера повідомлень.

3.3.1 Протоколи комунікації

Дана реалізація підтримує два протоколи комунікації в рамках агентної платформи: TCP та UDP (рисунок 3.7).

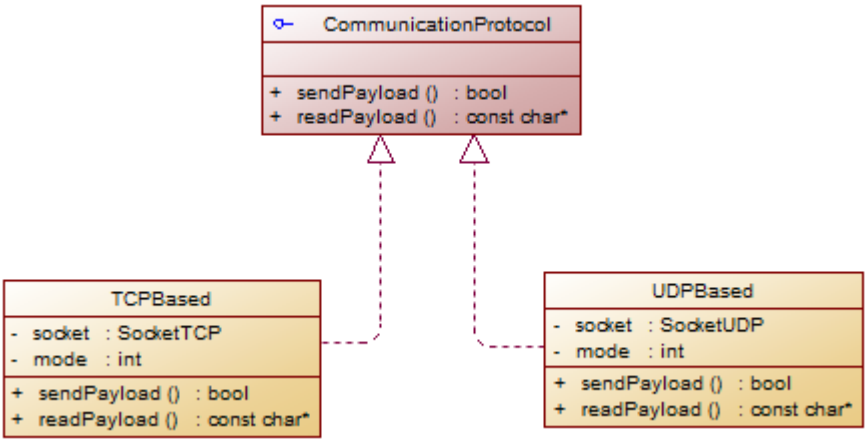

Рисунок 3.7 – протоколи комунікації.

Розробник агентної платформи може додати власні протоколи, передбачені стандартом FIPA (HTTP та ін.) або будь-яким іншим стандартом.

3.3.2 Протокол TCP

Протокол TCP надає транспортні послуги, що відрізняються від послуг UDP. Замість ненадійної доставки датаграм без встановлення з'єднань, він забезпечує гарантовану доставку з встановленням з'єднань у вигляді байтових потоків.

Протокол TCP використовується в тих випадках, коли потрібна надійна доставка повідомлень. Він звільняє прикладні процеси від необхідності використовувати тайм-аути і повторні передачі для забезпечення надійності. Великі можливості TCP даються не безкоштовно. Реалізація TCP вимагає великої продуктивності процесора і великої пропускної здатності мережі. Внутрішня структура модуля TCP набагато складніше структури модуля UDP. Прикладні процеси взаємодіють з модулем TCP через порти. Для окремих додатків виділяються загальновідомі номери портів.

Кожен агент містить пару TCP сокетів. Перший сокет функціонує у режимі читання. Адреса цього сокета є постійною й не змінюється протягом часу функціонування агента. Даний сокет отримує повідомлення та додає його у чергу. Повідомлення з черги обробляються послідовно та асинхронно за принципом FIFO (first in first out). Після цього з’єднання переривається і сокет готовий отримати нове повідомлення. Другий сокет працює у режимі запису. Під час запиту інформації або задачі агент надсилає відповідну команду за допомогою цього сокета. Дана реалізація дозволяє відмовитись від брокера повідомлень як від повноцінного агента. Принцип роботи показаний на рисунку 3.8.

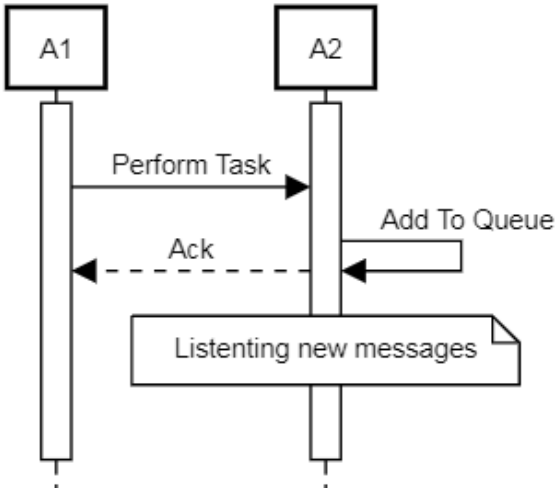

Рисунок 3.8 – принцип роботи брокера повідомлень.

Окрім пари сокеті, необхідних для брокера, кожен агент підіймає додаткові сокети при виконанні задач або при запитах інформації. Для кожної задачі агент створює один сокет, який приймає та відправляє повідомлення. Таким чином, повідомлення щодо задачі не потрапляють до основної черги. Даний підхід допомагає зменшити час виконання задачі та обробки результатів.

3.3.3 Протокол UDP

UDP використовує Інтернет-протокол для отримання дейтаграми (блоку даних) з одного комп'ютера на інший. UDP працює шляхом інкапсуляції даних у пакет UDP та додавання власної інформації заголовка до пакету. Ці дані складаються з вихідного порту та порту призначення, для комунікації, довжини пакета та контрольної суми. Після того, як пакети UDP будуть інкапсульовані у пакеті Інтернет-протоколів, вони відправляються до місця призначення.

На відміну від TCP, UDP не гарантує, що пакети потраплять у потрібні пункти призначення. Це означає, що UDP не підключається безпосередньо до приймального комп'ютера, як це робить TCP. Швидше, він надсилає дані і покладається на пристрої між комп’ютерами, що надсилають і приймають, щоб отримати дані там, де потрібно правильно відправитись.

Більшість програм, які використовують UDP, просто чекають будь-яких відповідей, які очікуються в результаті пакетів, відправлених через UDP. Якщо програма не отримує відповідь протягом певного періоду часу, програма знову надсилає пакет або він припиняє пробувати. UDP використовує просту модель передачі, яка не включає неявні діалоги рукостискань, щоб забезпечити надійність, замовлення або цілісність даних. Отже, послуга UDP є ненадійною, і пакети можуть вийти з ладу, схоже, що вони мають дублікати або зникають без попередження.

Хоча цей спосіб передачі не гарантує, що дані, які надсилаються, навіть досягнуть місця призначення, він має дуже низькі накладні витрати. Окрім цього, пакет може бути відправлений усім учасникам підмережі.

Кожен агент містить пару UDP сокетів. Перший сокет функціонує у режимі читання. Адреса цього сокета є постійною й не змінюється протягом часу функціонування агента та. Даний сокет отримує повідомлення додає його у чергу.

Повідомлення з черги обробляються послідовно та асинхронно за принципом FIFO (first in first out). Після цього з’єднання переривається і сокет готовий отримати нове повідомлення. Другий сокет працює у режимі запису. Агент передає свій статус під час запуску, після чого сокет звільнюється.

На відміну від TCP сокетів, UDP використовується лише для передачі стану агента під час старту. Даний підхід дозволяє відмовитись від постійних адрес основних агентів (брокеру повідомлень або служби каталогів). Недоліком є можливість втрати пакета. Якщо дані не були отримані через UDP, агент-приймач відправляє повторний запит за допомогою TCP, що гарантує успішну передачу інформації з другої спроби.

3.4 Мова комунікації агентів

Агент повинен мати модель представлення для представлення себе та свого погляду на світ. Модель представлення включає онтології, що представляють собою лексики та систематику, що визначають основні терміни та відносини в даній області, та мову змісту для представлення цих термінів та відносин. Наприклад, в області бездротового зв'язку агенти повинні мати словникові запаси для представлення різних компонентів (наприклад, еквалайзер, антена тощо); обробка (наприклад, вибірка, кодування / декодування, поширення частоти тощо) та цілі дій (наприклад, максимізація пропускної здатності або мінімізація перешкод). Враховуючи таку модель представлення, агентам також потрібна модель комунікації для фіксації комунікацій та потоку обміну знаннями в агентській спільноті. Модель зв'язку потребує набору мовних примітивів, які можна використовувати для реалізації моделі. Мова зв’язку агентів (ACL) загальновідома як мова, яка забезпечує набір мовних примітивів для реалізації моделі зв'язку агентів. ACL використовуються лише для побудови обгортки повідомлень і не стосуються вибору мови вмісту та моделі онтології.

Представлена мова комунікації агентів складається з 11 повідомлень. Базова структура повідомлення наведена на рисунку 3.9.

```
{
    agentInfo: { agentId: "id", agentType: "type" }
    addrInfo: { ip: "ip", port: "port" }
    taskInfo: { taskId: "id", taskType: "type" }
    additionalData: {}
}
```

Рисунок 3.9 – Базова структура повідомлення

Усі повідомлення передаються у форматі JSON. Кожне повідомлення містить інформацію про відправника, яка є унікальною для кожного агента. Також передається адреса, через яку комунікація може буди продовжена в разі необхідності додаткових даних.

3.4.1 UDP повідомлення

Мова комунікації агентів передбачає 3 типи повідомлень, які можуть буди передані засобами UDP протоколу: NotifyOn, NotifyOff, NotifyUnavailable.

Кожне з цих повідомлень передає стан агента та містить унікальну інформацію, що ідентифікує агента у мережі. Окрім цього, повідомлення може мати додаткову інформацію, таку як типи агентів, необхідних для подальшого виконання задач тощо.

Процеси реєстрації та дереєстрації агента у системі наведено на рисунку 3.10.

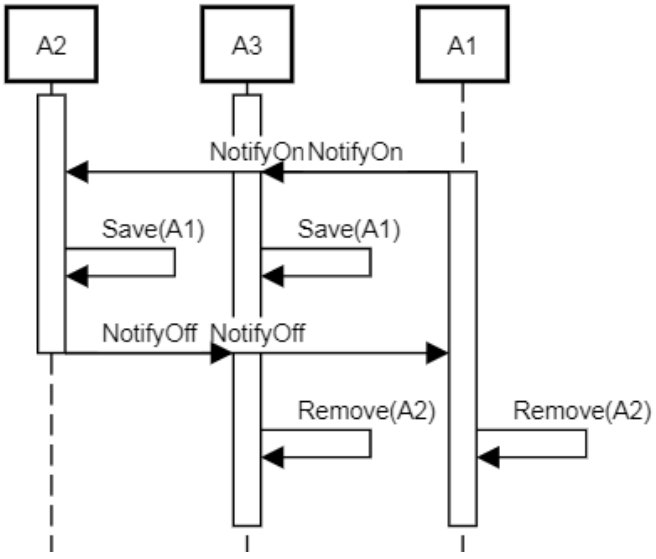

Рисунок 3.10 – процес реєстрація та дереєстрації агента у мережі

Повідомлення типу “NotifyOn” надсилається з метою реєстрації агента у мережі і містить наступну інформацію: унікальні ідентифікатора агента, тип задач, що виконує агент, адреса TCP сокету, який обробляє заявки на виконання задач,

масив типів агентів, що можуть бути необхідними для виконання задач даного агента.

Повідомлення типу “NotifyOff” надсилається з метою дереєстрації агента у мережі і містить наступну інформацію: унікальні ідентифікатора агента, тип задач, що виконує агент.

Повідомлення типу “NotifyUnavailable” надсилається з метою тимчасової дереєстрації агента у мережі і містить наступну інформацію: унікальні ідентифікатора агента, тип задач, що виконує агент, причина, з якої агент є тимчасово недоступним. Однією з причин може бути об’єднання агентів у групи, де функцію брокеру повідомлень виконує окремий агент. Детальніше даний підхід описаний у розділі 3.4.3.

3.4.2 TCP повідомлення

Мова комунікації агентів передбачає 6 типів повідомлень, які можуть буди передані засобами TCP протоколу: NotifyOnline, PerformTask, ConnectionRequest, DataRequest, DataResponse, TaskResult.

Кожне з цих повідомлень передає стан агента та містить унікальну інформацію, що ідентифікує агента у мережі. Окрім цього, повідомлення має інформацію про тип задачі та адресу, що може бути використана для подальшої комунікації. Принцип обміну повідомленнями наведено на рисунку 3.11.

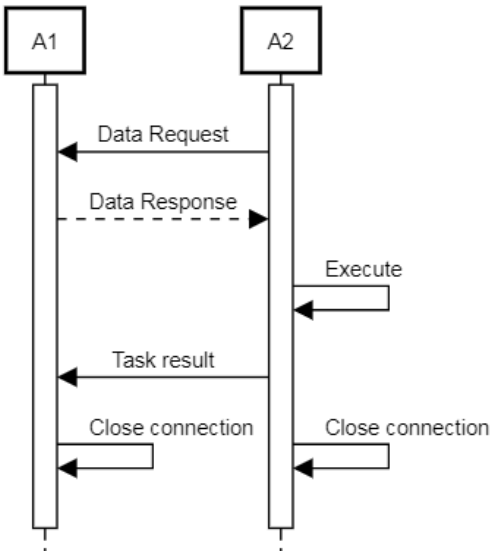

Рисунок 3.11 – принцип обміну TCP повідомленнями.

Повідомлення типу “NotifyOnline” є розширенням повідомлень статусу агента, описаних у розділі 3.4.1. Дане повідомлення надсилається у відповідь до повідомлення типу “NotifyOn” у випадку якщо відправник має потребу у інформації або послузі, що надає даний агент. Дане повідомлення містить унікальні ідентифікатори агента та адресу його брокера повідомлень.

Повідомлення типу “PerformTask” надсилається до агента, який надає інформацію або послугу. Дане повідомлення містить наступну інформацію: унікальні ідентифікатори агента, ідентифікатори задачі (тип), адресу, через яку може відбуватися подальша комунікація.

Повідомлення типу “DataRequest” надсилається у відповідь до повідомлення типу “PerformTask” у випадку, якщо агент потребує додаткових даних для виконання послуги. Дане повідомлення використовується з метою перевести комунікацію між брокерами агентів до відповідних каналів зв’язку.

Повідомлення типу “DataResponse” надсилається у відповідь до повідомлення типу “DataRequest” і містить інформацію, яку запитав агент-виконавець. Комунікація між агентами може бути перервана у результаті цього повідомлення, якщо агент-виконавець не надає результатів виконання задачі.

Повідомлення типу “TaskResult” надсилається агентом-виконавцем як результат запиту інформації чи послуги. Дане повідомлення містить ідентифікатори задачі, що була виконана, а також статус та результат виконання.

Повідомлення типу “ConnectionRequest” дане повідомлення може бути використане замість повідомлення типу “PerformTask” у випадку, коли агент потребує окремого каналу зв’язку для подальшої комунікації щодо виконання задачі. Повідомлення такого типу містить унікальні ідентифікатори агента-відправника, адресу для подальшої комунікації, а також причину для встановлення з’єднання. Агент-отримувач може відмовити у комунікації відповідно до причини.

3.4.3 Додаткові типи повідомлень

Мультиагентна система може складатися з агентів подібних за типом виконуваних задач. З метою оптимізації ресурсів такі агенти можуть бути об’єднані у групи з введенням додаткового агента у систему. Функціонування агента в рамках

групи описується трьома додатковими повідомленнями: NotifyNewTask, PickTask, TaskToExecute.

Приклад поведінки агентів у групі наведено на рисунку 3.12.

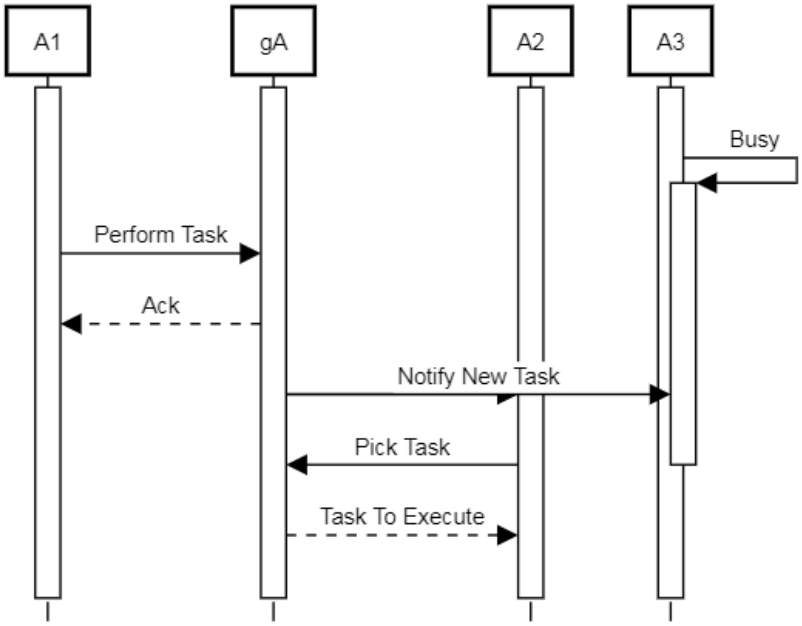

Рисунок 3.12 – Поведінка агентів у групі.

Повідомлення типу “NotifyNewTask” надсилає агент-провайдер до усіх агентів групи. Дане повідомлення містить унікальні ідентифікатори агента-провайдера.

Повідомлення типу “PickTask” надсилає агент учасник групи у відповідь до повідомлення типу “NotifyNewTask”, якщо даний агент вільний і може виконати задачу. Дане повідомлення містить унікальні ідентифікатори агента-учасника та інформацію про його адресу.

Повідомлення типу “TaskToExecute” надсилає агент-провайдер у відповідь до повідомлення типу “NotifyNewTask”, у випадку, якщо він має задачі у черзі на виконання. Дане повідомлення містить інформацію про агента-провайдера, а також оригінальне повідомлення-запит на виконання задачі.

Агенти, які об’єднані у групи, можуть функціонувати у рамках групи та індивідуально. Таким чино вихід із ладу агента-провайдера групи не завдасть шкоди мультиагентній системі.

3.4.4 Можливості розширення мови

Модуль опису мови комунікації агентів складається з трьох основних частин, перевизначення яких дозволяє розширити дану мову новими типами повідомлень.

Кожне повідомлення імплементує інтерфейс “Serializable”. Даний інтерфейс використовується для перетворення повідомлення у формат для пересилання. Приклад наведено на рисунку 3.13

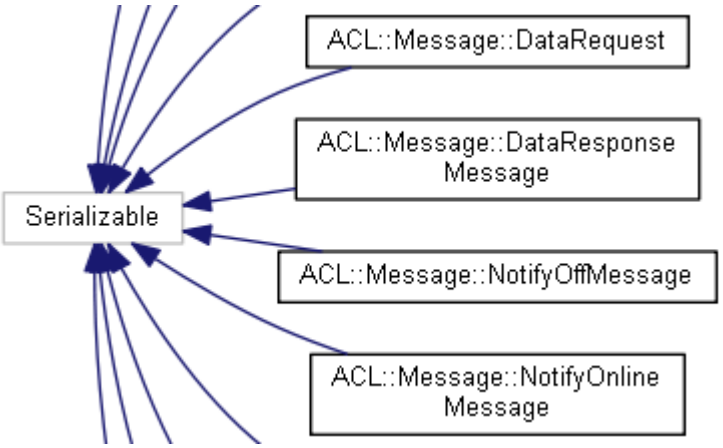

Рисунок 3.13 – класи повідомлень

Розробник агента може додати власне повідомлення і описати його структуру за таким самим принципом.

Мова комунікації описує принципи обробки повідомлень за допомогою окремих класів. Кожен клас імплементує інтерфейс “Handler”. Даний підхід дозволяє зберігати чергу базових класів, що зменшує зв’язність коду. Приклад наведено на рисунку 3.14.

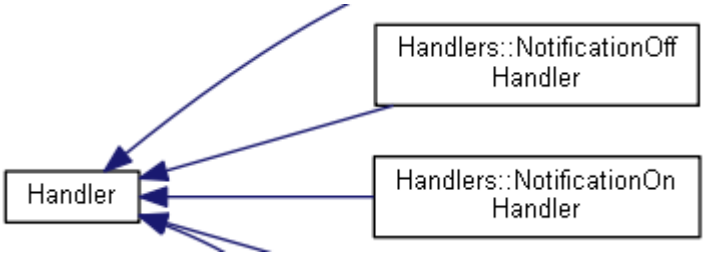

Рисунок 3.14 – класи обробки повідомлень

Розробник агента повинен створити клас-обробник власного повідомлення за аналогічним принципом. При цьому логіка обробки повідомлення може бути задана розробником і є незалежною від системи.

Класи-обробники реєструються у системі через шаблон проектування “Фабрика”. Під час отримання повідомлення ACL визначає тип і знаходить відповідний клас-обробник через абстрактну фабрику. Приклад зв’язку обробників і фабрики наведено на рисунку 3.15.

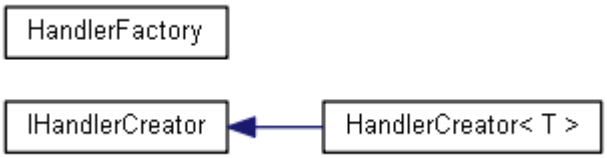

Рисунок 3.15 - зв’язок обробників і фабрики

Під час реалізації власних повідомлень розробник агента повинен зареєструвати клас-обробник свого повідомлення у глобальній фабриці обробників мови комунікації. Мова ACL надає публічне API для даних цілей.

3.5 Тестування та документація

Модульне тестування, або юніт-тестування (Unit testing) - процес в програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми. Ідея полягає в тому, щоб писати тести для кожної нетривіальною функції або методу. Це дозволяє досить швидко перевірити, чи не призвело чергову зміну коду до регресії, тобто до появи помилок в уже оттестировать місцях програми, а також полегшує виявлення і усунення таких помилок.

Програмний код має набір юніт-тестів, що перевіряють правильність роботи базових функцій, таких як: геренація токенів, збереження даних у пам’яті, отримання класів обробників з глобальної фабрики тощо. Покриття коду юніт-тестами дозволило підвищити швидкість виявлення регресій під час розробки нового функціоналу.

Не усі модулі можуть бути протестовані за допомогою модульних тестів. Модулі транспортної служби, які відповідають за протоколи зв’язку між агентами залежать від системи, яку використовує агент. В рамках цього було додано інструментальні тести платформи Android. Інструментальні модульні тести - це тести, які працюють на фізичних пристроях та емуляторах, і вони можуть скористатися API-програмами Android. Інструментальні тести забезпечують більшу надійність, ніж звичайні юніт-тести, але вони працюють набагато повільніше. Даний підхід надає можливість протестувати роботу сокетів у мережі.

Задля автоматизації запуску тестів та автоматичного знаходження регресій між змінами використано концепцію безперервної інтеграції (Continuous Integration) та безперервної доставки (Continuous Delivery) (CI/CD). Принцип дії CI / CD схожий на конвеєр: інструмент виконує інтеграційну функцію, включаючи різні типи автоматичних тестів на кожному етапі, з наступною доставкою та розгортанням завершеного коду в готовий продукт для кінцевого користувача.

3.6 Кросплатформеність

Кросплатформеність - здатність програмного забезпечення функціонувати в декількох різних операційних системах або на різних апаратних платформах. З метою досягнення кросплатформеності програмний код має відповідні модулі, які використовуються на основних платформах Windows, Linux, Android та MacOS. Окрім цього, розробник агента може додати підтримку нових платформ перевизначивши протоколи комунікації.

Компіляція проекту також можлива на платформах Windows та Linux. Це досягається за допомогою використання інструменту CMake. CMake - це сімейство кросплатформених інструментів, призначених для створення, тестування та пакетування програмного забезпечення. CMake використовується для управління процесом компіляції програмного забезпечення за допомогою простих файлів конфігурації, незалежних від платформи та компілятора, та генерування власних файлів і робочих просторів, які можна використовувати в обраному середовищі.

У рамках роботи перевірено компіляцію проекту за допомогою наступних компіляторів: MSVC (Windows), G++ (Linux, MacOS), ndk-tools (Android).

Роботу агентів перевірено у наступних середовищах: Windows, Android.

3.7 Висновки до розділу 3

Даний розділ розкриває основні архітектурні особливості реалізації кожного компоненту та системи в цілому.

Використання архітектурних шаблонів зробило розробку швидкою, а результат - гнучким до розширення. Кожен компонент агенту може бути протестований окремо, без створення відповідних зв’язків з іншими компонентами, що робить тестування швидким та надійним.

4. СТАРТАП ПРОЕКТ

4.1 Опис ідеї проекту

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
надати можливість розробникам створювати системи на основі мульиагентного підходу з мінімальними затратами ресурсів	Платний, оплата здійснюється разово.	Пришвидшення розробки систем, витрати меншої кількості ресурсів на розробку системи.

Відрізняється від існуючих аналогів та замінників можливістю розробки додатків на декількох платформах та для декількох платформ, додаток покриває більшу частину найпоширеніших платформ. Монетизація відбувається за рахунок оплати за доступ до продукту. Присутні наступні версії:

- Розширена (одноразова оплата) – повний функціонал без обмежень у часі.
- Підписка на підтримку – додається можливість отримувати нові версії продукту, комунікувати з розробниками з метою виявлення проблем.
- Інтеграція зі сторонніми сервісами.

Користувач може керувати підпискою, а саме: призупиняти час дії 1 раз за рік.

Кооперація із зовнішніми сервісами відбувається шляхом переговорів з власниками в індивідуальному порядку. Власнику стороннього сервісу надається пакет підписок зі зниженими цінами з метою залучення нових клієнтів до даного додатку.

4.1.1 Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів.

Copernic Agent (http://www.copernic.com/) - одночасно відправляє запити кільком популярним пошуковим системам, вибирає найбільш рейтингові посилання, зіставляє їх між собою, видаляє дублі і, сортуючи відібране по рейтингу відповідно до свого алгоритму ранжирування, виводить їх користувачеві.

MySimon (http://www.mysimon.com/) - здійснює інтелектуальний пошук, порівнюючи ціни мільйонів товарів в більш ніж двох тисячах онлайнових магазинів.

MP3-Wolf (http://www.trellian.com/) - сканує Інтернет в пошуках потрібних користувачеві музичних файлів. У процесі роботи він використовує різні пошукові системи, а також сайти, знайдені ним раніше і що містяться в його базі.

WebSite-Watcher (http://www.aignes.com/) - призначена для стеження за змінами на сайтах. Підтримує роботу RSS-стрічки. Має гнучкі настройки щодо запобігання помилкових спрацьовувань, коли окремі зміни на сторінках носять випадковий або технічний характер, наприклад зміна числа переглядів.

Таблиця 4.2 – порівняння конкурентних компаній.

Назва	Платформа	Аудиторія	Ціна
Copernic Agent	Web, Android, iOS	США	Наявні різні види підписок
MySimon	Web, Android, iOS	США, Великобританія	Безкоштовно, розширена версія – 49$
MP3-Wolf	Web, Android, iOS	США, Європа	Безкоштовно
WebSite-Watcher	Web	США, Індія	Безкоштовно
Даний проект	Windows, Linux, Android, MacOS	Європа	Наявні різні види підписок

Кожна з конкуруючих компаній покриває хоча б одну з найпоширеніших платформ (web, Android, iOS) та орієнтується на клієнтів з Європи або США.

Основним доходом компаній, які надають свої додатки безкоштовно, є реклама. Іншим можливим варіантом доходу є оплата підписки на користування протягом певного періоду або наявність розширеної версії з більшою кількісті функцій.

Додатки кожної з конкуруючих компаній мають свій набір функцій, але він не є повністю унікальним. Тому порівняємо додатки за найбільш вживаними функціями. Порівняння наведені у таблиці 4.3.

Таблиця 4.3 – порівняння функціоналу.

Фунцкія	Copernic Agent	MySimon	WebSite-Watcher	Даний проект
Можливість модифікації існуючих агентів	+	-	-	+

Продовження таблиці 4.3

Розробка агентів	+	+-	+	+
Можливість виключення агентів з системи	-	-	+	+
Можливість роботи системи з 1 агентом	+	+	+	+
Можливість модифікації структури системи	-	+	-	+

4.2 Технологічний аудит ідеї проекту

Таблиця 4.4. Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Інтерфейс користувача	Розробка кросплатформеного PWA	є	є
2	Реклама для перегляду	Google Ads	є	є
3	Збереження та обробка даних, захист цих даних	Використання баз даних	є	є
4	Система оплати підписок	Інтеграція з PayPal тощо	є	є
5	Захист даних під час передачі	SSL сертифікат, використання HTTPS	є	є
6	Системи координації	Алгоритми пріоритизації	є	є

Висновок: проект можливо реалізувати з технічної точки зору, усі потрібні технології існують та частково надаються безкоштовно.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Даний підрозділ аналізує можливий попит на даний продукт та в цілому описує можливості запуску даного прокету.

4.3.1 Аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку

Таблиця 4.5. Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	Більше 10
2	Загальний обсяг продаж	$7.6 трлн
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Технологічний аудит продукьу
5	Специфічні вимоги до стандартизації та сертифікації	Необхідно забезпечити легальність контенту
6	Середня норма рентабельності в галузі (або по ринку), %	15

Висновок: ринок розробки СДК для використання як основи мультиагентних систем є привабливим для входження за попереднім оцінюванням.

4.3.2 Потенційні групи клієнтів

Таблиця 4.6. Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Вимоги споживачів до товару
1	Використання мультиагентних систем	Кінцевий користувач	Якість, надійність, швидка технічна підтримка
2	Організація агентів у групи	Розробник додатку	Якість, надійність, швидка технічна підтримка
3	Модифікації структури системи	Розробник додатку, власник додатку	Якість, надійність, швидка технічна підтримка

4.3.3 Фактори, що сприяють ринковому впровадженню проекту, та фактори, що йому перешкоджають.

Основні фактори перелічені у таблицях 4.7 та 4.8.

Таблиця 4.7. Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Фінансування	Проблема збитковості проекту на початковій стадії інтеграції	Пошук інвесторів, кредит у банку (Приват / Моно)
2	Сторонні сервіси	Проблема укладання договорів	Переговори, зміна сервісу.

Продовження таблиці 4.7

3	Популярність сервісу	Недостатня активність користувачів та поширеність на ринку	Реклама, SEO
4	Захист даних	Можливі проблеми з довірою користувачів до ресурсу через незнання реалізації.	Реклама, інформація у додатку, що стосується захисту даних
5	Безпека даних	Викрадення даних користувачів	Орендувати систему захисту даних
6	Надійність	Збої у роботі під час «часів пік»	Орендувати сервер на умовах у дата центрі з умовою підтримки.
7	Розробка додатку	Можливі складнощі при розробці відео сервісу, відставання від термінів тощо	Збільшити команду розробників
8	Інтеграція та супровід	Можливі непередбачені проблеми	Найняти команду технічного суповоду.

Таблиця 8. Фактори можливостей

№	Фактор	Зміст можливостей	Можлива реакція компанії
1	Прибуток	Отримання прибутку	Заходи для збільшення рентабельності
2	Розширення ринку	Покриття додаткових потреб користувачів	Додати інтеграцію зі сторонніми сервісами
3	Розширення аудиторії	Збільшення кількості активних користувачів	Забезпечити користувачів необхідним функціоналом

4.3.4 Аналіз пропозиції.

Таблиця 4.9. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: олігополія 1-го роду	Існує певна кількість додатків зі схожим функціоналом	Забезпечити користувачів більшою кількістю функцій

Продовження таблиці 4.9

2. За рівнем конкурентної боротьби - глобальний	Додатки-аналоги доступні в більшості країн світу	Локалізація інтерфейсу додатку, оренда серверів у різних частинах світу.
3. За галузевою ознакою -міжгалузева	Можлива конкуренція між повністю безкоштовними та платними додатками, між додатками та туристичними фірмами	Підвищення якості надання послуг
4. Конкуренція за видами товарів: - товарно-родова	Можлива конкуренція між повністю безкоштовними та платними додатками, між додатками та туристичними фірмами	Підвищення комфорту використання додатку (переробка UI/UX).
5. За характером конкурентних переваг - цінова та нецінова	Цінова – кількість функцій за певну оплату Нецінова - якість та доступність послуг	Цінова – більша кількість послуг за ту саму ціну Нецінова – цілодобова підтримка користувачів та серверів
6. За інтенсивністю - не марочна	Конкуренція за надання послуг планування подорожі	Реклама

4.3.5 Аналіз конкуренції.

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі за моделлю 5 сил М. Портера (таблиця 4.10).

Таблиця 4.10. Аналіз конкуренції в галузі за М. Портером

Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари замінники
Copernic Agent, MySimon, WebSite-Watcher	Бар’єри входження в ринок: Гнучкі ціни Розмір капіталовкладень Зв’язки із власниками сторонніх сервісів	Фактори сили постачальників: Змінні витрати постачальників	Фактори сили: Змінні витрати Високий рівень чутливості до зміни цін Контроль якості	Фактори загроз: Ціна Лояльність споживачів

Продовження таблиці 4.10

Висока інтенсивність конкурентної боротьби	Є можливості входу в ринок Є потенційні конкуренти Строки виходу їх на ринок 0.5-1 рік	Сторонні сервіси надають свої послуги на обмеженій за територіальною або національною приналежністю користувача.	Важливе відношення ціна / набір функцій / якість.	Обмеження для роботи на ринку через товари замінники: обмеження економічної рентабельності

Висновок: проект буде конкурентним за умови реалізації наступних функцій: можливість зміни структури мультиагентної системи, можливість модифікації агентів, можливість відмови від агентів середнього рівня.

4.3.6 Обґрунтування факторів конкурентоспроможності

Таблиця 4.11. Обґрунтування факторів конкурентоспроможності

№	Фактор конкуренто спроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Гнучка система підписок	Можливість «заморожувати» підписку за бажанням на певний строк, отримання потрібного функціоналу в залежності від типу підписки.
2	Модифікація структури системи	Можливість вводу та виведення агентів до(з) структури МАС.
3	Інтеграція сторонніх сервісів	Розробник може додати власні сервіси до існуючої архітектури МАС.

4.3.7 Аналіз сильних та слабких сторін стартап-проекту.

Таблиця 4.12. Порівняльний аналіз сильних та слабких сторін.

№	Фактор конкурентоспроможності	Рейтинг у порівнянні з конкурентами
1	Гнучка система підписок	+1
2	Можливість вилучення агентів середнього рівня	+2
3	Можливість зміни структури системи	+3

Загальний рейтинг у порівнянні з конкурентами складає +6 одиниць.

4.3.8 SWOT- аналіз стартап-проекту

Таблиця 4.13. SWOT- аналіз стартап-проекту

Сильні сторони: Можливість вилучення агентів середнього рівня Можливість зміни структури системи за бажанням розробника	Слабкі сторони: Часткова залежність від сторонніх сервісів. Фінансування Популярність сервісу
Можливості: Потенційна прибутковість. Швидка інтеграція та популяризація. Швидкий розвиток ринку.	Загрози: Кількість сильних конкурентів. Низька кваліфікація команди розробників (якість продукту).

4.3.9 Альтернативи ринкового впровадження стартап-проекту

Таблиця 4.14. Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Замовлення розробки та підтримки у сторонніх компаній	Фінансові витрати середні, початок отримання доходу у середині строку реалізації.	4-6 місяців
2	Формування своєї команди розробки та підтримки	Значні фінансові витрати, особливо на початкових етапах	6-8 місяців
3	Розробка проекту власними силами на початковому етапі	Відсутність потреби у фінансуванні.	11-14 місяців.

Висновок: обрано комбінацію варіантів №1 та № 2, початковий етап виконує стороння компанія, з часом формується власна команда розробки, що зменшує частку витрат на сторонню компанію.

4.4 Розроблення ринкової стратегії проекту

4.4.1 Стратегія охоплення ринку

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку та опис цільових груп потенційних споживачів (таблиця 4.15).

Таблиця 4.15. Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Сторонні сервіси	низька	низький	висока	важко
2	Користувач	висока	середній	висока	середня

Було обрано цільові групи: власники сторонніх сервісів, користувачі.

Стратегія охоплення ринку - стратегія диференційованого маркетингу (працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу).

4.4.2 Базова стратегія розвитку

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 4.16).

Таблиця 4.16. Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Стратегія зростання	масовий маркетинг	Нижча вартість за набір базових функцій	Стратегія лідерства по витратах
2	Стратегія стабілізації	стратегія диференційованого маркетингу	Користувач – відношення ціна / якість / функціонал Власник стороннього сервісу – популяризація.	Стратегія диференціації

В якості базової стратегії було обрано **стратегію диференціації.**

4.4.3 Вибір стратегії конкурентної поведінки.

Таблиця 17. Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	ні	так	Так, частина базового функціоналу.	Стратегія виклику лідера

4.4.4 Стратегія позиціонування

Стратегія позиціонування полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.18. Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Цінова доступність	Наступальна стратегія	Безкоштовний додаток за рахунок вбудованої реклами	Ціна, якість та безпека
2	Можливість керування контентом (реклами та безпосередньо відео контент)	Наступальна стратегія	Гнучка система керування підписками	

4.5 Розроблення маркетингової програми стартап-проекту

4.5.1 Маркетингова концепція товару

Визначимо ключові переваги концепції використання запропонованої архітектури МАС (рисунок 4.19).

Таблиця 4.19. Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Планування подорожі	Інструментарій для планування, інтеграція зі сторонніми сервісами	-Гнучка система вибору способу оплати -Гнучка система керування підписками -Кооперація груп у часі та просторі
2	Інтеграція сторонніх сервісів	Збільшення прибутків за рахунок розширення аудиторії	
3	Кооперація груп туристів	Функціонал створення та адміністрування груп.	

4.5.2 Трирівнева маркетингова модель товару.

Таблиця 4.20. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Товар – це допоміжник у подорожі. Товар надає можливість планування подорожі одному або групою.
2. Товар у реальному виконанні	Властивості: Ціна, грн або інша валюта Режим оплати (безкоштовний з рекламою, платний, підписка). Розмір, Гб Швидкість передачі даних, Мбіт/с. Опис
3. Товар із підкріпленням	Графічний інтерфейс користувача з локалізацією.
За рахунок чого потенційний товар буде захищено від копіювання: обфускація коду додатку, регулювання на рівні законодавства.	

4.5.3 Цінові межі

Визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар.

Таблиця 4.21. Визначення меж встановлення ціни

№	Рівень цін на товари замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	30-60$	0-60$	всі	0$, 60$

4.5.4 Визначення оптимальної системи збуту.

Таблиця 4.22. Формування системи збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Індивідуальне та групове споживання товарів та послуг	Збут здійснюється власними силами через створений мережу Інтернет	Однорівневий канал	«Сторонній сервіс – додаток - користувач»

4.5.5 Концепція маркетингових комунікацій

Розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку.

Таблиця 4.23. Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Власники сторонніх сервісів потребують індивідуального підходу для укладання угод із прийнятними для них умовами.	Інтернет, бізнес зустрічі	Персональний продаж, прямий маркетинг	Зацікавити власників сторонніх сервісів співпрацювати з відео сервісом для отримання прибутку	Переконати власників сторонніх сервісів, що співпраця з проектом дозволить розширити аудиторію користувачів, що призведе до збільшення їх прибутків
Розробник потребує можливості створювати мультиагентні системи на основі існуючих стандартів	Інтернет	Реклама	Зацікавити розробників використовувати даний сервіс	Відношення ціна / якість / функціонал

4.6 Менеджмент ризиків

Для визначення основних ризиків використаємо ментальні карти. Загальний вигляд інтелектуальної карти управління ризиками наведено на рисунку 4.1.

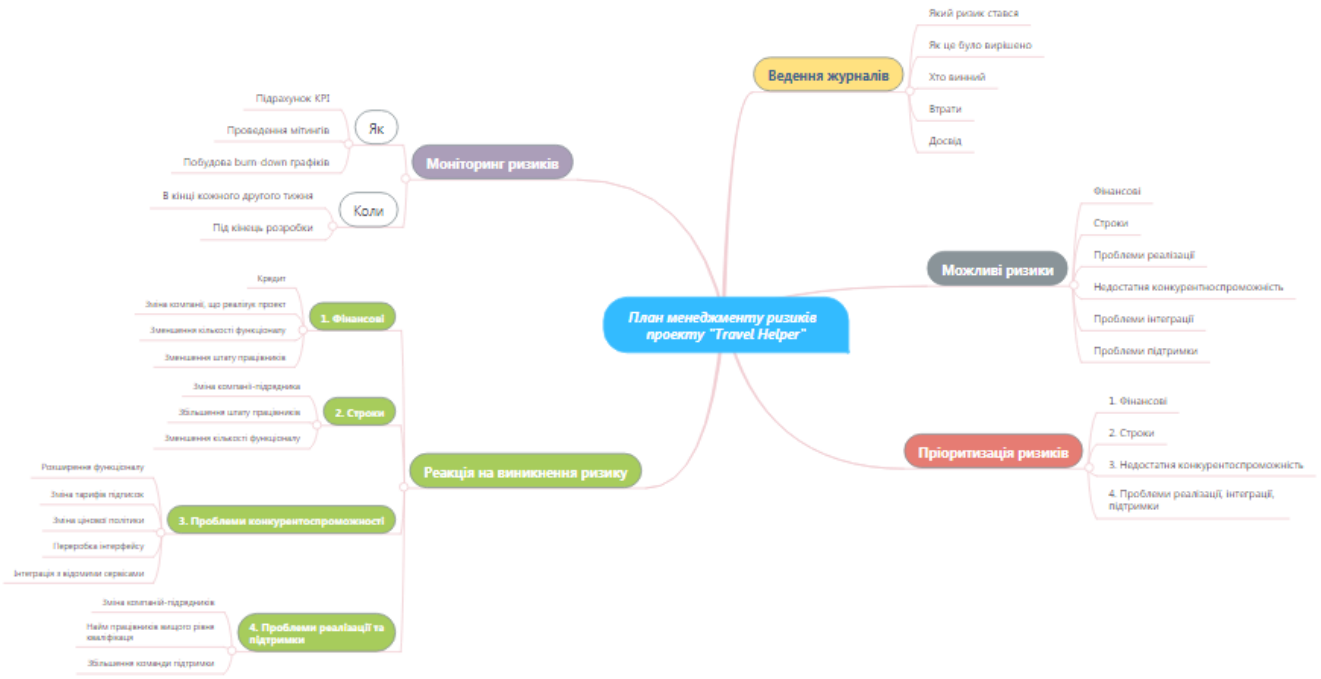

Рисунок 4.1 – Ментальна карта управління ризиками

Розглянемо основні рівні ментальної карти як області для ідентифікації можливих ризиків (рисунок 4.2)

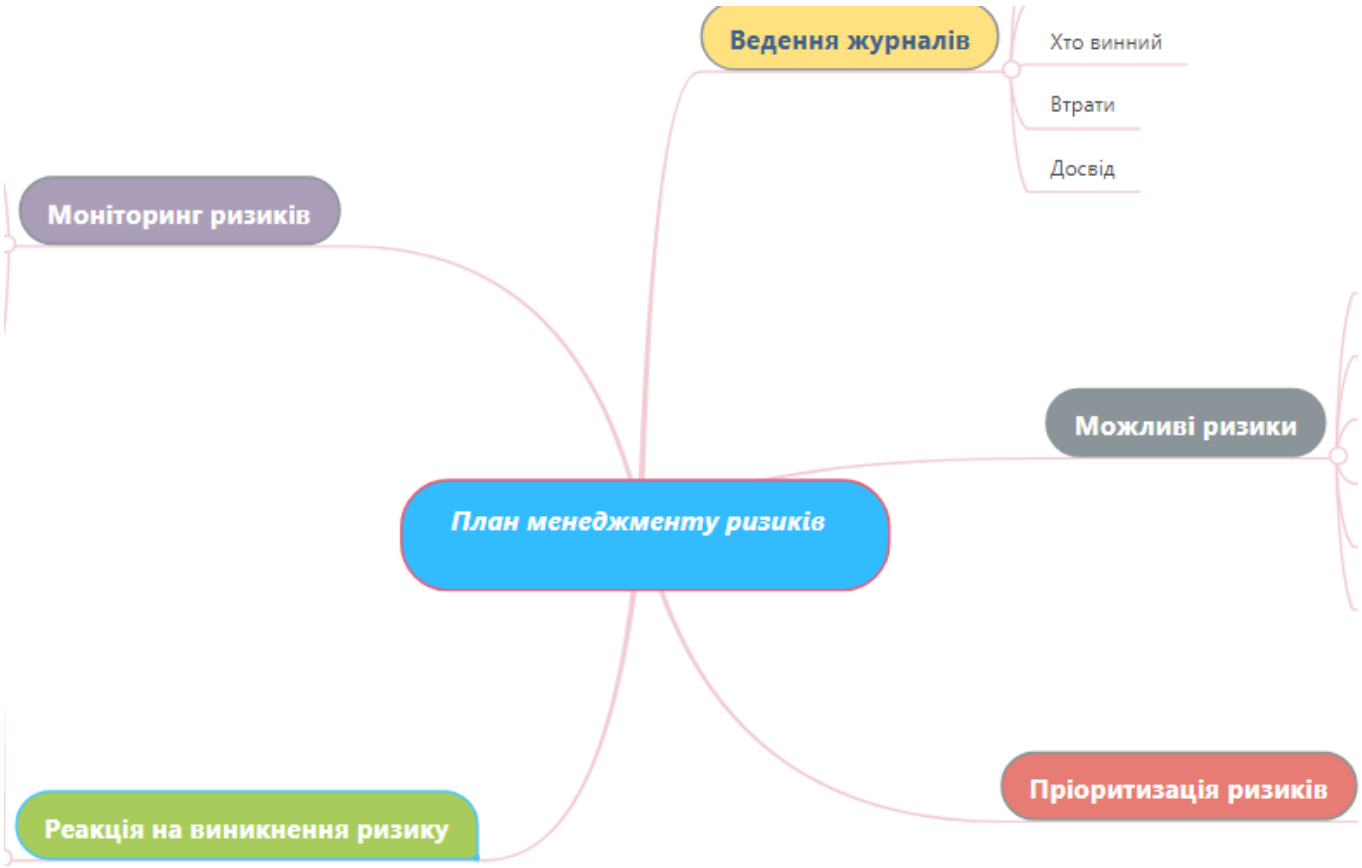

Рисунок 4.2 – основні рівні ментальної карти ризиків.

Розглянемо детальніше перші дві сфери ідентифікації ризиків (рисунок 4.3).

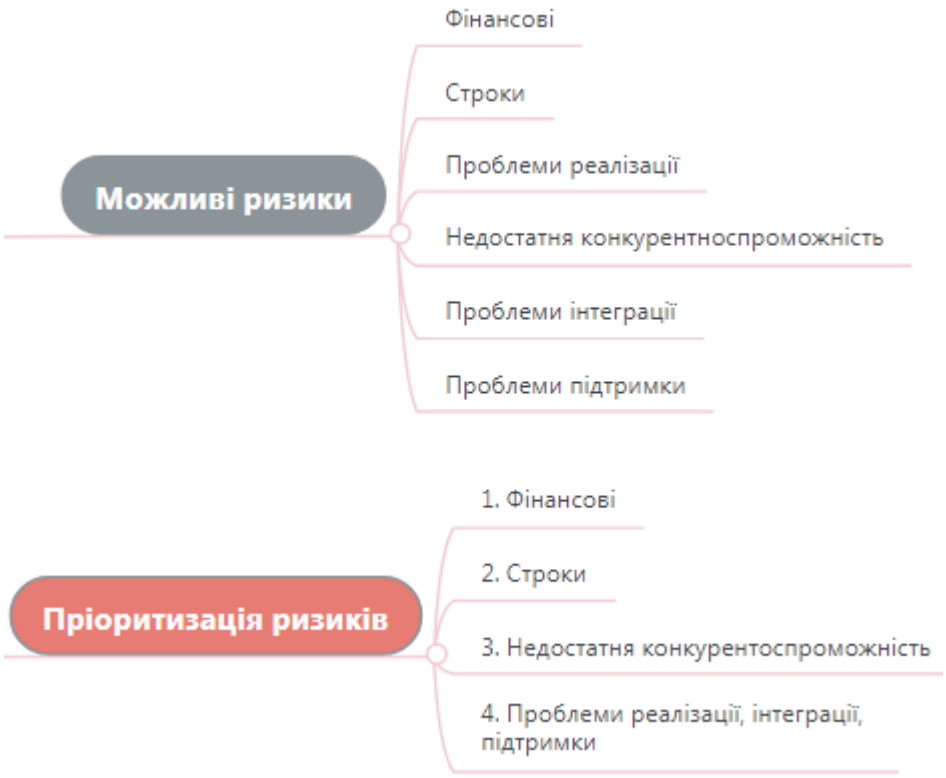

Рисунок 4.3 – Можливі ризики та їх пріоритети.

Розглянемо можливі реакції на виникнення ризиків, перелічених вище (рисунок 4.4) та способи їх моніторингу (рисунок 4.5).

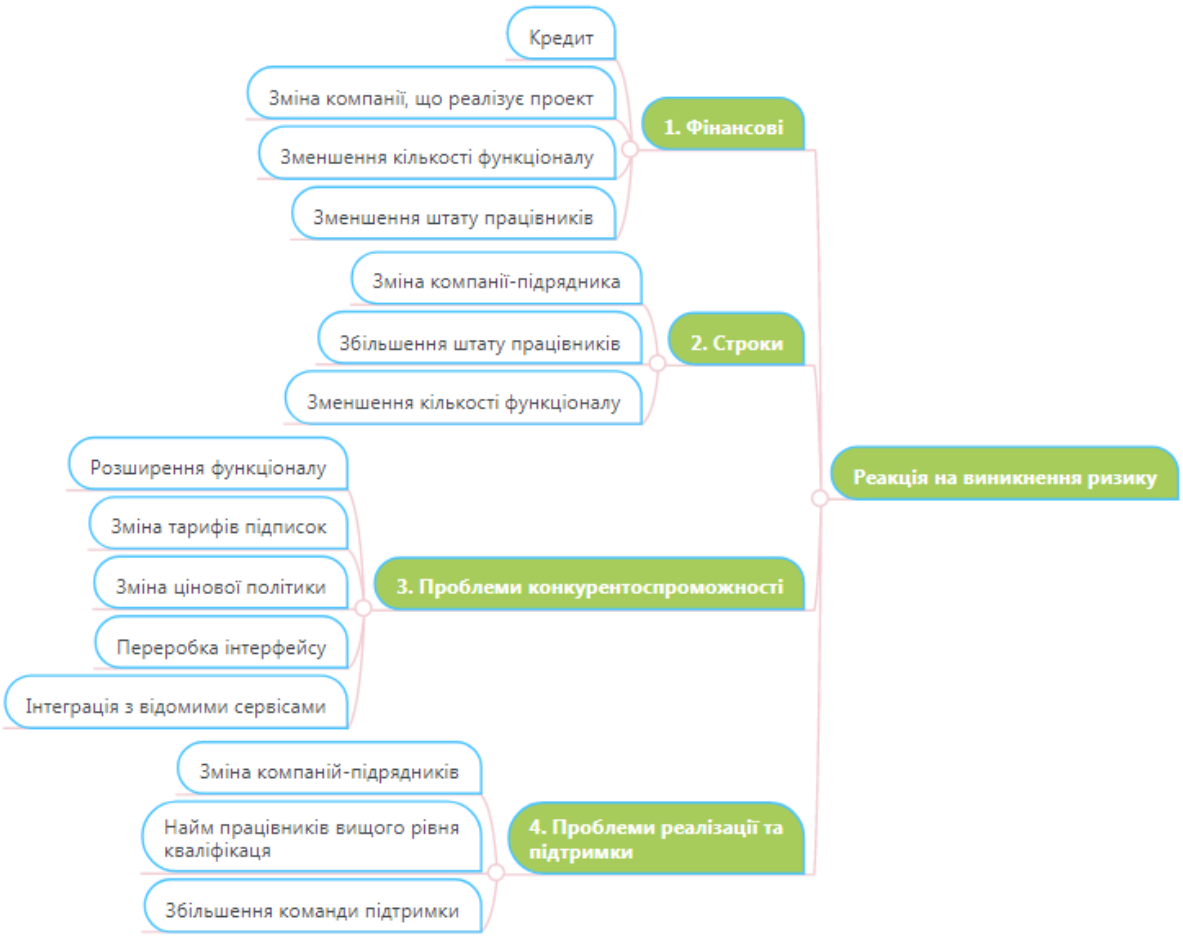

Рисунок 4.4 – Реакція на виникнення ризиків.

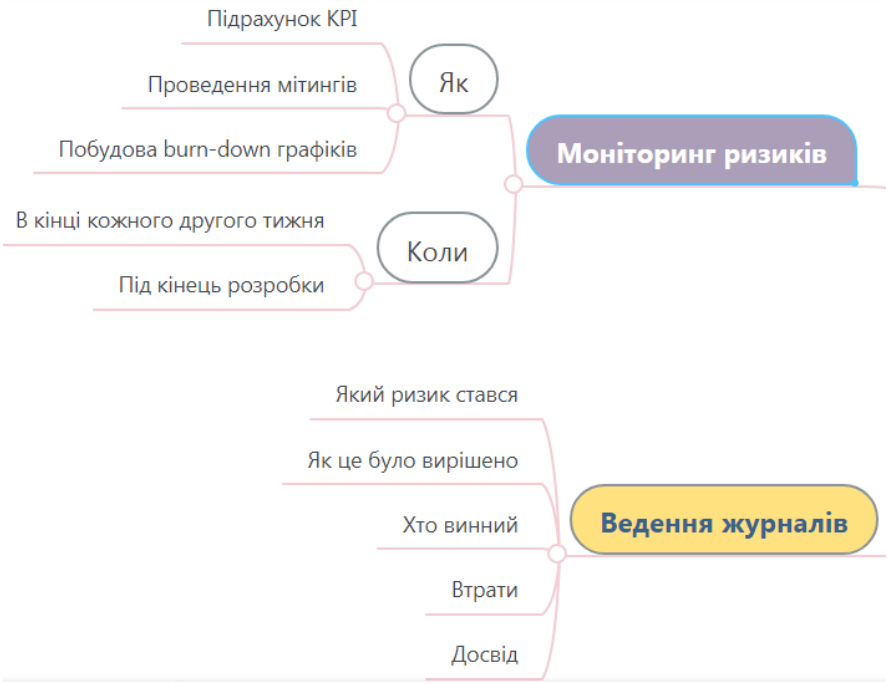

Рисунок 4.5 – Моніторинг ризиків.

Даний підрозділ показує можливість використання ментальних карт як гнучкого інструменту для ідентифікації ризиків та способів управління ними.

4.7 Висновки до розділу 4

Було проведено аналіз ідеї додатку для розробки мультиагентних систем та зроблено висновки:

- чи є можливість ринкової комерціалізації проекту - так;
- чи є перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження, стан конкуренції, конкурентоспроможність проекту – так;
- яку альтернативу (варіант) впровадження доцільно обрати для ринкової реалізації проекту - розробка проекту силами сторонньої компанії на початковому етапі, формування власної команди розробки та підтримки;
- чи є доцільною подальша імплементація проекту - так.

ВИСНОВКИ

Основний результат дисертаційної роботи полягає в підвищенні якості розроблюваних мультиагентних систем. Підвищення якості МАС, розроблених із застосуванням запропонованих моделей, досягається за рахунок зменшення цикломатичної складності, використання загальних кількостей елементів системи, збільшення ефективності та управління, адаптації та адаптивності системи, шляхом порівняння з мультиагентними системами, побудованими з використанням стандартизованої архітектури FIPA.

Запропонована архітектурна програмна реалізація дозволила зменшити кількість залежних між компонентами. Розроблені інтерфейси підтримують інтеграцію сторонніх компонентів, що підвищує прикладну цінність системи.

В результатах дослідження:

- Проаналізували існуючих архітектур на відповідність критеріям якості мультиагентних систем;
- Виявили проблеми розробки мультіагентних систем реального часу;
- Запропоновані моделі та алгоритми організовані у фреймворк, що зменшує затрати часу на розробку моделей агентів;

Таким чином, у виконанні диссертаційної роботи були зроблені всі поставлені завдання та досягнуті цілі дослідження.

Перспективні результати розвитку тематичних досліджень:

1. Дослідження та розробка алгоритмів машинного навчання для адаптації функцій оцінювання пріоритетних цілей агентів у процесі моделювання;

2. Розробка моделей взаємодії агентів для більшого ефективності виконання оцінок;

3. Розвиток фреймворку та розширення наборів програмних компонентів.

4. Уніфікація механізму безпеки агентів.

Спроектована мультиагентна система дозволяє істотно спростити впровадження сучасних систем автоматизації технологічної підготовки виробництва за рахунок їх більш тісної інтеграції, а також знизити накладні витрати, пов'язані з

промисловою експлуатацією подібних систем. Запропоновані методи можуть бути адаптовані для вирішення технологічних задач не тільки в приладобудуванні, але і в суміжних галузях, таких як медицина, логістика, системи безпеки (контроль доступу).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IoT Middleware: A Survey on Issues and Enabling Technologies, IEEE Internet Things / A. H. Ngu, M. Gutierrez, V. Metsis та ін.]., 2017. – 1 с.

2. A.Medvedev. Interoperability and Open-Source Solutions for the Internet of Things / A. Medvedev., 2017. – 157 с.

3. M. A. G. Maureira. ThingSpeak – an API and Web Service for the Internet of Things / M. A. G. Maureira, L. Teernstra., 2011.

4. ThingWorx Industrial Innovation Platform | PTC. [Електронний ресурс] – Режим доступу до ресурсу: https://www.thingworx.com/platforms/.

5. Open-Source IoT Platform 2017 — IoT cloud platform the Internet of Things solutions and applications that set the standard. [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaaproject.org/platform/.

6. M. El Beqqal. Access control system in campus combining RFID and biometric based smart card technologies / M. El Beqqal, M. A. Kasmi, M. Azizi., 2017. – 569 с. – (Adv. Intell. Syst. Comput.).

7. J. Chhabra. IoT based Smart Home Design using Power and Security Management / J. Chhabra. // Iciccs. – 2016. – С. 6–10.

8. A.Chianese. Designing a smart museum: When cultural heritage joins IoT / A. Chianese, F. Picciali., 2014. – 310 с. – (8th Int. Conf. Next Gener. Mob. Appl. Serv. Technol. NGMAST).

9. Leo Louis. Working Principle of Arduino and using IT as a tool for study and Research / Leo Louis. // International Journal of Control, Automation Communication and System (IJCAS). – 2016. – №1. – С. 2.

10. Raspberry PI and Arduino UNO Working Together as a Basic Meteorological Station / J. R. Cortes Leon, R. F. Martínez-Gonzalez, A. M. Medina, L. A. Peralta-Pelaez. // International Journal of Computer Science and Information Technology. – 2017. – №5. – С. 97–104.

11. M. Petracca. AMBER: Advanced Mother Board for Embedded systems pRototyping / M. Petracca, P. Passaro,, E. Gioia. // EURASIP Journal on Embedded Systems. – 2017. – №1.

12. IoT Middleware: A Survey on Issues and Enabling Technologies / A. H. Ngu, M. Gutierrez, V. Metsis та ін.]. // IEEE Internet Things J.. – 2017. – №4. – С. 1–20.

13. U. of California Berkeley. TinyDB: A Declarative Database for Sensor Networks. [Електронний ресурс] / U. of California Berkeley – Режим доступу до ресурсу: http://telegraph.cs.berkeley.edu/tinydb/.

14. Open Platforms – Global Sensor Networks (GSN) [Електронний ресурс] – Режим доступу до ресурсу: http://open-platforms.eu/library/global-sensor-networks-gsn/.

15. Скобелев П. О. Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений Автометрия / Скобелев П. О.. – 2002. – №6. – С. 45–61.

16. Амелин К.С. Применение мультиагентного подхода для решения задач мониторинга местности группой легких БПЛА / Амелин К.С., Граничин О.Н. // сб. тр. межд. научно- практ. конф. Теория активных систем. – 2011. – №3. – С. 209–214.

17. Городецкий В.И. Многоагентные системы (обзор). / Городецкий В.И., Грушинский М.С., Хабалов А.В.. // Новости искусственного интеллекта.. – 1998. – №2. – С. 64–116.

18. Nwana. H. Software agents: An overview / Nwana. H. // Knowledge and Engineering Review. – 1996. – №11.

19. Commercial applications of agents: lessons, experiences and challenges / R. A. Belecheanu, S. Munroe, M. Luck та ін.]. // 2006. – С. 1549–1555.

20. M. Sierhuis. Nasas oca mirroring system an application of multiagent systems in mission control / M. Sierhuis, W. J. Clancey. // AAMAS. – 2009. – С. 85–92.

21. Iris a tool for strategic security allocation in transportation networks / J. Tsai, S. Rathi, C. Kiekintveld та ін.]. // AAMAS. – 2009. – С. 85–92.

22. J. Himoff. Magenta technology: multiagent systems for industrial logistics / J. Himoff, P. Skobelev, M. Wooldridge. // AAMAS. – 2005. – С. 60–66.

23. G. Caire. Wade: a software platform to develop mission critical applications exploiting agents and workflows / G. Caire, D. Gotta, M. Banzi. // AAMAS. – 2008. – С. 29–36.

24. K. Dorer. An adaptive solution to dynamic transport optimization / K. Dorer, M. Calisti. // AAMAS. – 2005. – С. 45–51.

25. S. Jacobi. Masdispoxt: heat and sequence optimisation based on simulated trading inside the supply chain of steel production / S. Jacobi, D. Raber, K. Fischer. // AAMAS. – 2008. – С. 23–26.

26. Multi-agent planning in mass-oriented production / M. Pechoucek, M. Rehak, P. Charvat та ін.]. // IEEE Transactions System, Man and Cybernetics. – 2007. – №3. – С. 386–395.

27. V. Darley. An agent-based model of a corrugated box factory: The tradeoff between finished-goods- stock and on-time-in-full delivery / V. Darley, P. V. Tessin, D. Sanders. // Fifth Workshop on AgentBased Simulation. – 2004.

28. S. S. Benfield. Making a strong business case for multiagent technology / S. S. Benfield, J. Hendrickson, D. Galanti. // AAMAS. – 2006. – С. 10–15.

29. PSL: A Semantic Domain for Flow Models [Електронний ресурс] – Режим доступу до ресурсу: https://www.nist.gov/publications/psl-semantic-domain-flow-models.

30. Cyc KB [Електронний ресурс] – Режим доступу до ресурсу: https://www.cyc.com/products2.html.

ДОДАТОК А

Публікації

Децентралізація функцій служби каталогів мультиагентної системи енергетичної інфраструктури з метою підвищення горизонтальної масштабованості та стабільності системи

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ41161_19М

Аркушів 4

2019

www.konferenciaonline.org.ua

Міжнародна наукова інтернет-конференція

Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення

(випуск 43)

Частина 1

ISSN 2522-932X

14 листопада 2019 р.

Тернопіль
2019

Література:
1. *ВВЭР-1000*: физические основы эксплуатации, ядерное топливо, безопасность / Афров А.М. та ін. Москва, 2006, 488 с.
2. *АЭС* с реактором типа ВВЭР-1000. От физических основ эксплуатации до эволюции проекта / Андрушечко С.А. та ін. Москва, 2010, 604 с.
3. *Медведєв Р.Б.* Функція приналежності для алгоритму виявлення водяної пари у теплоносії першого контуру АЕС з реактором ВВЕР-1000 / Р.Б. Медведєв, Д.М. Складанний, А.Р. Крайнік // Збірник наукових статей КМХТ 2019, Київ, 2019, С. 111-112.

Міхєєв О.С.
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ
Кафедра автоматизації проектування енергетичних процесів і систем, студент

ДЕЦЕНТРАЛІЗАЦІЯ ФУНКЦІЙ СЛУЖБИ КАТАЛОГІВ МУЛЬТИАГЕНТНОЇ СИСТЕМИ ЕНЕРГЕТИЧНОЇ ІНФРАСТРУКТУРИ З МЕТОЮ ПІДВИЩЕННЯ ГОРИЗОНТАЛЬНОЇ МАСШТАБОВАНОСТІ ТА СТАБІЛЬНОСТІ СИСТЕМИ

Служба каталогів - це компонент мультиагентної платформи, який надає агентам послуги з жовтими сторінками. Основною метою служби каталогів є надійне збереження інформації щодо стану агентів активної системи, а також підтримання точного, повного та актуального переліку агентів. Тобто, служба каталогів надає найактуальнішу інформацію про будь-якого агента свого каталогу на недискримінаційній основі всім уповноваженим агентам.

Кожен агент, який хоче оприлюднити свої послуги іншим агентам, повинен знайти відповідну службу каталогів та запросити реєстрацію його опису. Для реєстрації необхідно надати опис агента, що містить обов'язкові параметри опису. Також можуть бути надані необов'язкові та приватні параметри, що містять стандартизовану інформацію, яку розробник агента вважає необхідною до збереження у каталозі. Не передбачено майбутніх зобов'язань чи зобов'язань з боку зареєстрованого агента. Наприклад, агент може відмовити в запиті на виконання функції, яка зареєстрована через «Службу каталогів». Крім того, служба каталогів не може контролювати життєвий цикл будь-якого з зареєстрованих агентів.

Наявність каталогу агентів і єдиного головного контейнера платформи дозволяє агентам спілкуватися (обмінюватися інформацією) між собою. При цьому така взаємодія можливо всередині контейнера, між контейнерами однієї платформи (між агентами різних контейнерів на одній платформі) і між платформами (між агентами в контейнерах різних платформ). Обмін повідомленнями здійснюється асинхронно, з використанням черг.

Служба каталогів є найбільш навантаженим агентом мультиагентної системи через постійні звернення за інформацією з боку інших учасників системи. Стандарт FIPA передбачає наявність декількох служб каталогів з

метою зменшення навантаження на кожен з каталогів. Такий підхід збільшує час обробки інформації з метою прийняття рішень з боку функціональних агентів, що може призвести до непередбачуваних наслідків у рамках енергетичної інфраструктури.

Одним із варіантів розподілу навантаження на службу каталогів є децентралізація функцій реєстрації та дереєстрації між агентами-учасниками системи. Кожен агент містить інформації про свої функціональні можливості та локально зберігає інформацію лише про необхідних йому агентів або служб. Використання протоколу UDP дозволяє зменшити залежність між агентами та обмінюватись інформацією про актуальний стан системи без встановлення з'єднання за допомогою широкомовних повідомлень. Для передачі подібних повідомлень між локальними підмережами або глобальними мережами є можливість організувати агент типу проксі, метою та єдиною функцією якого буде передача будь-яких повідомлень між агентами подібного типу.

Даний підхід вирішує наступні проблеми: зменшення часу на прийняття рішення за рахунок збільшення швидкості отримання інформації про агента-виконавця, зменшення залежності агент-сервіс за рахунок використання UDP повідомлень, що передбачає також зменшення витрат на організацію та підтримку системи, зменшує необхідність у вертикальному масштабуванні системи адже кожен агент одночасно є сервісом каталогів.

Література:

1. FIPA Agent Management Specification [Електронний ресурс] – Режим доступу до ресурсу: http://www.fipa.org/specs/fipa00023/SC00023K.html#_Toc75950983.
2. Postel J. INTERNET STANDARD [Електронний ресурс] / Postel. – 1980. – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc768.

Ніколаєнко Р.С.,
Національний Технічний Університет України "Київський політехнічний інститут ім. Ігоря Сікорського", м. Київ
Кафедра автоматики та управління в технічних системах, студент

ЗАХИЩЕНА P2P КОМУНІКАЦІЯ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN

На сьогодні обмін повідомленнями є найбільш використовуваною мережевою функцією, і аутентифікація між користувачами є життєво важливою властивістю. Найпоширенішими підходами до забезпечення цієї властивості є протоколи шифрування PKI та S/MIME, але вони вразливі до атак MITM та атаки EFAIL. Блокчейн - це інноваційна технологія, яка долає ці загрози і дозволяє децентралізувати чутливі операції, зберігаючи високий рівень безпеки. Це усуває необхідність для довірених посередників. Блокчейн доступний для всіх мережевих вузлів і відслідковує всі вже здійснені транзакції. Блокчейн пропонує модель дизайну повідомлень, що забезпечують ефективність та